Lecture 7: Clustering

Information Retrieval

Computer Science Tripos Part II

Ronan Cummins[1]

Natural Language and Information Processing (NLIP) Group

UNIVERSITY OF CAMBRIDGE

ronan.cummins@cl.cam.ac.uk

2016

[1]Adapted from Simone Teufel's original slides

Upcoming

Upcoming

- What is clustering?

Upcoming

- What is clustering?
- Applications of clustering in information retrieval

Upcoming

- What is clustering?
- Applications of clustering in information retrieval
- K-means algorithm

Upcoming

- What is clustering?
- Applications of clustering in information retrieval
- K-means algorithm
- Introduction to hierarchical clustering

Upcoming

- What is clustering?
- Applications of clustering in information retrieval
- K-means algorithm
- Introduction to hierarchical clustering

Upcoming

- What is clustering?
- Applications of clustering in information retrieval
- K-means algorithm
- Introduction to hierarchical clustering

Upcoming

- What is clustering?
- Applications of clustering in information retrieval
- K-means algorithm
- Introduction to hierarchical clustering
- Single-link and complete-link clustering

Overview

Clustering: Definition

Clustering: Definition

- (Document) clustering is the process of grouping a set of documents into clusters of similar documents.

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.

- (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.

- (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.

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.

Difference clustering–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 from centroid ("degree of cluster membership")

The cluster hypothesis

The cluster hypothesis

Cluster hypothesis.

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

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.

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.

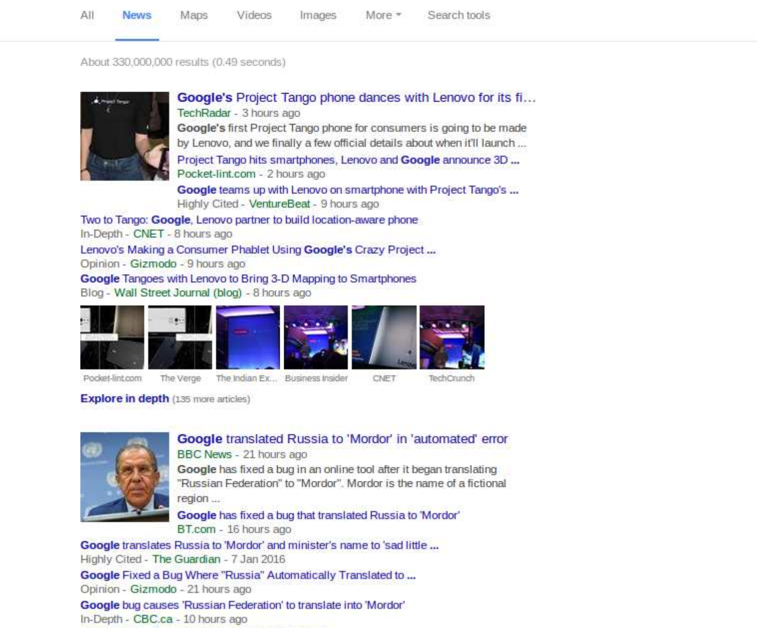
Van Rijsbergen's original wording (1979): "closely associated documents tend to be relevant to the same requests".

Applications of Clustering

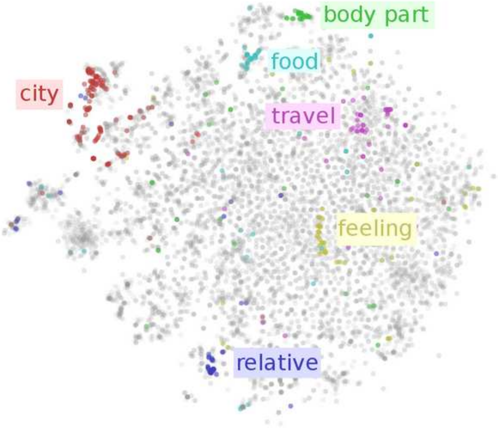
- IR: presentation of results (clustering of documents)
- Summarisation:
  - clustering of similar documents for multi-document summarisation
  - clustering of similar sentences for re-generation of sentences
- Topic Segmentation: clustering of similar paragraphs (adjacent or non-adjacent) for detection of topic structure/importance
- Lexical semantics: clustering of words by cooccurrence patterns

Clustering search results

Clustering news articles



Clustering Words



2

[2] https://colah.github.io/posts/2015-01-Visualizing-Representations/

Types of Clustering

- Hard clustering v. soft clustering
  - Hard clustering: every object is member in only one cluster
  - Soft clustering: objects can be members in more than one cluster
- Hierarchical v. non-hierarchical clustering
  - Hierarchical clustering: pairs of most-similar clusters are iteratively linked until all objects are in a clustering relationship
  - Non-hierarchical clustering results in flat clusters of "similar" documents

Desiderata for clustering

Desiderata for clustering

- General goal: put related docs in the same cluster, put unrelated docs in different clusters.

Desiderata for clustering

- General goal: put related docs in the same cluster, put unrelated docs in different clusters.
  - We'll see different ways of formalizing this.

Desiderata for 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.

Desiderata for 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.

Desiderata for 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.
  - There also exist semiautomatic methods for determining K

Desiderata for 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.
  - There also exist semiautomatic methods for determining K
- Secondary goals in clustering

Desiderata for 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.
  - There also exist semiautomatic methods for determining K
- Secondary goals in clustering
  - Avoid very small and very large clusters

Desiderata for 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.
  - There also exist 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

Desiderata for 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.
  - There also exist 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
  - Many others ...

Overview

Non-hierarchical (partitioning) clustering

- Partitional clustering algorithms produce a set of k non-nested partitions corresponding to k clusters of n objects.
- Advantage: not necessary to compare each object to each other object, just comparisons of objects – cluster centroids necessary
- Optimal partitioning clustering algorithms are $O(kn)$
- Main algorithm: K-means

K-means: Basic idea

- Each cluster j (with n_j elements x_i) is represented by its centroid c_j, the average vector of the cluster:

$$c_j = \frac{1}{n_j} \sum_{i=1}^{n_j} x_i$$

K-means: Basic idea

- Each cluster j (with n_j elements x_i) is represented by its centroid c_j, the average vector of the cluster:

$$c_j = \frac{1}{n_j} \sum_{i=1}^{n_j} x_i$$

- Measure of cluster quality: minimise mean square distance between elements x_i and nearest centroid c_j

$$RSS = \sum_{j=1}^{k} \sum_{x_i \in j} d(\vec{x_i}, \vec{c_j})^2$$

K-means: Basic idea

- Each cluster j (with n_j elements x_i) is represented by its centroid c_j, the average vector of the cluster:

$$c_j = \frac{1}{n_j} \sum_{i=1}^{n_j} x_i$$

- Measure of cluster quality: minimise mean square distance between elements x_i and nearest centroid c_j

$$RSS = \sum_{j=1}^{k} \sum_{x_i \in j} d(\vec{x_i}, \vec{c_j})^2$$

- Distance: Euclidean; length-normalised vectors in VS

K-means: Basic idea

- Each cluster j (with n_j elements x_i) is represented by its centroid c_j, the average vector of the cluster:

$$c_j = \frac{1}{n_j} \sum_{i=1}^{n_j} x_i$$

- Measure of cluster quality: minimise mean square distance between elements x_i and nearest centroid c_j

$$RSS = \sum_{j=1}^{k} \sum_{x_i \in j} d(\vec{x_i}, \vec{c_j})^2$$

- Distance: Euclidean; length-normalised vectors in VS
- We iterate two steps:

K-means: Basic idea

- Each cluster j (with n_j elements x_i) is represented by its centroid c_j, the average vector of the cluster:

$$c_j = \frac{1}{n_j} \sum_{i=1}^{n_j} x_i$$

- Measure of cluster quality: minimise mean square distance between elements x_i and nearest centroid c_j

$$RSS = \sum_{j=1}^{k} \sum_{x_i \in j} d(\vec{x_i}, \vec{c_j})^2$$

- Distance: Euclidean; length-normalised vectors in VS
- We iterate two steps:
  - reassignment: assign each vector to its closest centroid

K-means: Basic idea

- Each cluster j (with n_j elements x_i) is represented by its centroid c_j, the average vector of the cluster:

$$c_j = \frac{1}{n_j} \sum_{i=1}^{n_j} x_i$$

- Measure of cluster quality: minimise mean square distance between elements x_i and nearest centroid c_j

$$RSS = \sum_{j=1}^{k} \sum_{x_i \in j} d(\vec{x_i}, \vec{c_j})^2$$

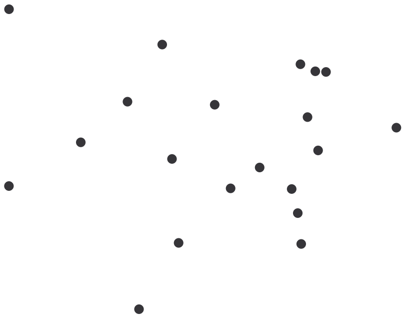
- Distance: Euclidean; length-normalised vectors in VS
- We iterate two steps:
  - reassignment: assign each vector to its closest centroid
  - recomputation: recompute each centroid as the average of the vectors that were recently assigned to it

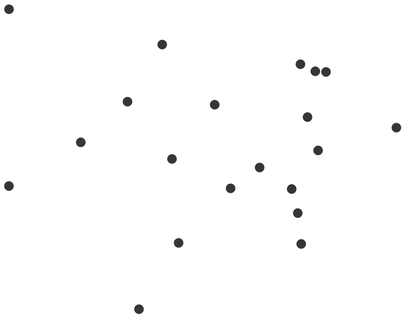
K-means algorithm

Given: a set $s_0 = \overrightarrow{x_1}, \ldots \overrightarrow{x_n} \subseteq \mathcal{R}^m$
Given: a distance measure $d : \mathcal{R}^m \times \mathcal{R}^m \rightarrow \mathcal{R}$
Given: a function for computing the mean $\mu : \mathcal{P}(\mathcal{R}) \rightarrow \mathcal{R}^m$

Select k initial centers $\overrightarrow{c_1}, \ldots \overrightarrow{c_k}$
while stopping criterion not true:
$\sum_{j=1}^{k} \sum_{x_i \in s_j} d(\overrightarrow{x_i}, \overrightarrow{c_j})^2 < \epsilon$ (stopping criterion)
do
for all clusters s_j **do** *(reassignment)*
$c_j := \{\overrightarrow{x_i} | \forall \overrightarrow{c_l} : d(\overrightarrow{x_i}, \overrightarrow{c_j}) \leq d(\overrightarrow{x_i}, \overrightarrow{c_l})\}$
end
for all means $\overrightarrow{c_j}$ **do** *(centroid recomputation)*
$\overrightarrow{c_j} := \mu(s_j)$
end
end

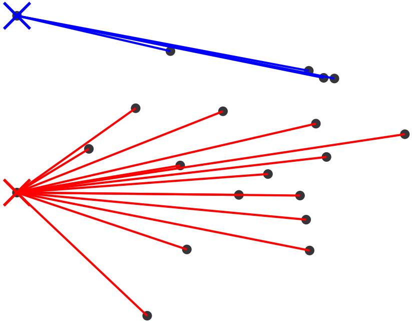
Worked Example: Set of points to be clustered





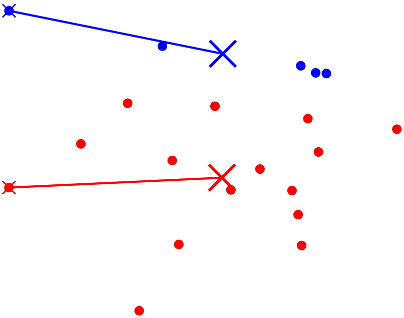
Exercise: (i) Guess what the optimal clustering into two clusters is in this case; (ii) compute the centroids of the clusters

Random seeds + Assign points to closest center



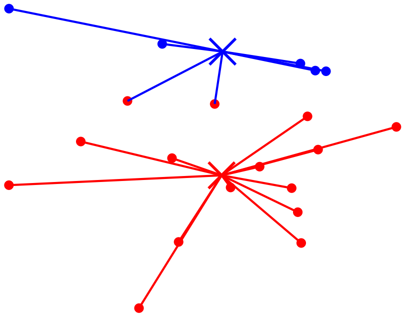
Iteration One

Worked Example: Recompute cluster centroids



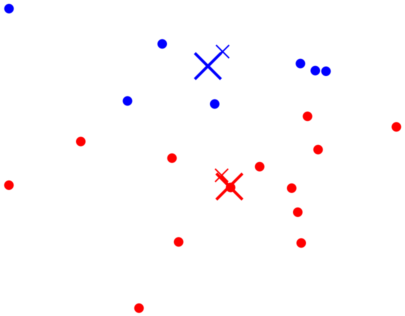
Iteration One

Worked Example: Assign points to closest centroid

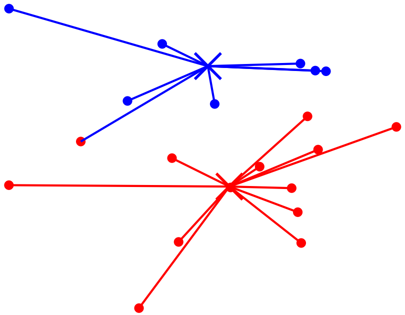


Iteration One

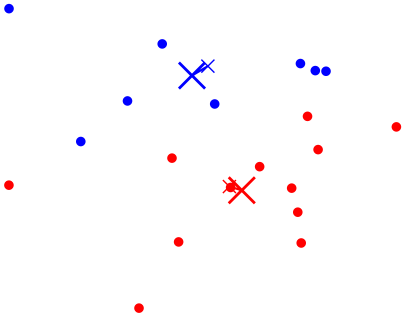
Worked Example: Recompute cluster centroids



Iteration Two

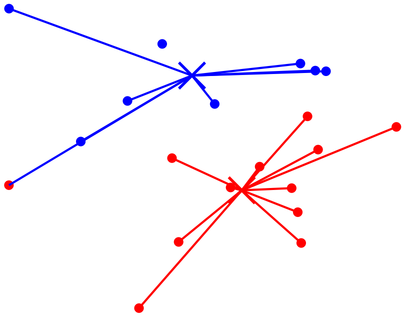


Iteration Two



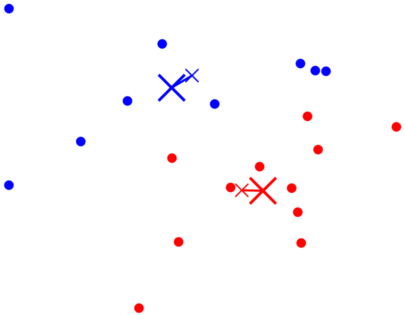
Iteration Three

Worked Example: Assign points to closest centroid



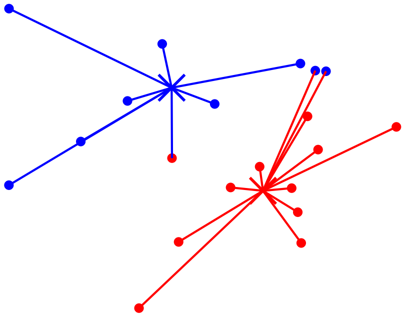
Iteration Three

Worked Example: Recompute cluster centroids



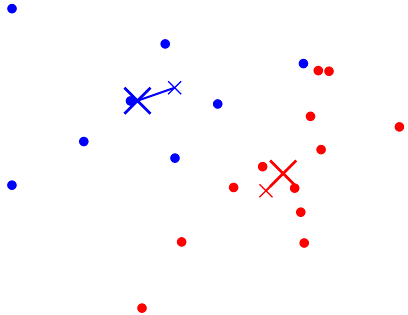
Iteration Four

Worked Example: Assign points to closest centroid

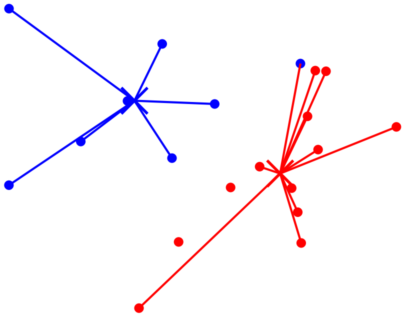


Iteration Four

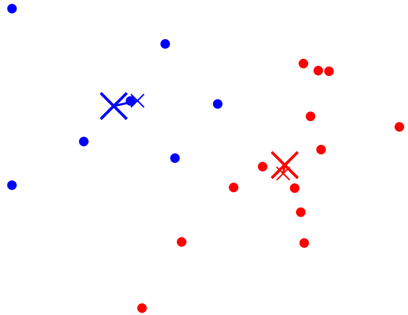
Worked Example: Recompute cluster centroids



Iteration Five

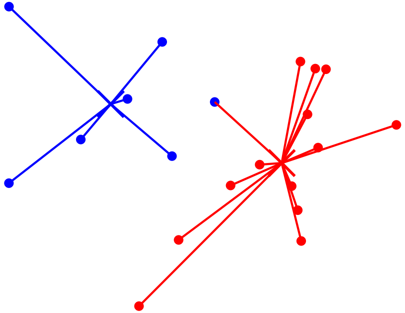


Iteration Five



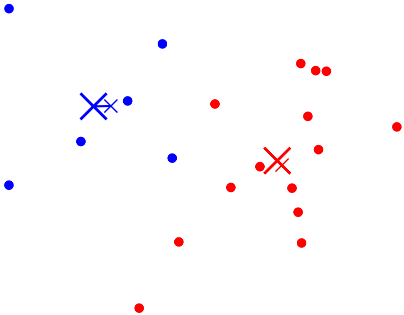
Iteration Six

Worked Example: Assign points to closest centroid

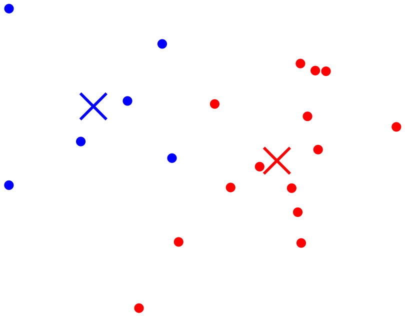


Iteration Six

Worked Example: Recompute cluster centroids



Iteration Seven



Convergence

K-means is guaranteed to converge: Proof

K-means is guaranteed to converge: Proof

- RSS decreases during each reassignment step.

K-means is guaranteed to converge: Proof

- RSS decreases during each reassignment step.
  - because each vector is moved to a closer centroid

K-means is guaranteed to converge: Proof

- RSS decreases during each reassignment step.
  - because each vector is moved to a closer centroid
- RSS decreases during each recomputation step.

K-means is guaranteed to converge: Proof

- RSS decreases during each reassignment step.
  - because each vector is moved to a closer centroid
- RSS decreases during each recomputation step.
  - This follows from the definition of a centroid: the new centroid is the vector for which RSS_k reaches its minimum

K-means is guaranteed to converge: Proof

- RSS decreases during each reassignment step.
  - because each vector is moved to a closer centroid
- RSS decreases during each recomputation step.
  - This follows from the definition of a centroid: the new centroid is the vector for which RSS_k reaches its minimum
- There is only a finite number of clusterings.

K-means is guaranteed to converge: Proof

- RSS decreases during each reassignment step.
  - because each vector is moved to a closer centroid
- RSS decreases during each recomputation step.
  - This follows from the definition of a centroid: the new centroid is the vector for which RSS_k reaches its minimum
- There is only a finite number of clusterings.
- Thus: We must reach a fixed point.

K-means is guaranteed to converge: Proof

- RSS decreases during each reassignment step.
  - because each vector is moved to a closer centroid
- RSS decreases during each recomputation step.
  - This follows from the definition of a centroid: the new centroid is the vector for which RSS_k reaches its minimum
- There is only a finite number of clusterings.
- Thus: We must reach a fixed point.
- Finite set & monotonically decreasing evaluation function $\rightarrow$ convergence

K-means is guaranteed to converge: Proof

- RSS decreases during each reassignment step.
  - because each vector is moved to a closer centroid
- RSS decreases during each recomputation step.
  - This follows from the definition of a centroid: the new centroid is the vector for which RSS_k reaches its minimum
- There is only a finite number of clusterings.
- Thus: We must reach a fixed point.
- Finite set & monotonically decreasing evaluation function $\rightarrow$ convergence
- Assumption: Ties are broken consistently.

Other properties of K-means

- Fast convergence
    - K-means typically converges in around 10-20 iterations (if we don't care about a few documents switching back and forth)
    - However, complete convergence can take many more iterations.
- Non-optimality
    - K-means is not guaranteed to find the optimal solution.
    - If we start with a bad set of seeds, the resulting clustering can be horrible.
- Dependence on initial centroids
    - Solution 1: Use i clusterings, choose one with lowest RSS
    - Solution 2: Use prior hierarchical clustering step to find seeds with good coverage of document space

Time complexity of K-means

Time complexity of K-means

- Reassignment step: $O(KNM)$ (we need to compute KN document-centroid distances, each of which costs $O(M)$

Time complexity of K-means

- Reassignment step: $O(KNM)$ (we need to compute KN document-centroid distances, each of which costs $O(M)$
- Recomputation step: $O(NM)$ (we need to add each of the document's $< M$ values to one of the centroids)

Time complexity of K-means

- Reassignment step: $O(KNM)$ (we need to compute KN document-centroid distances, each of which costs $O(M)$
- 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

- Reassignment step: $O(KNM)$ (we need to compute KN document-centroid distances, each of which costs $O(M)$
- 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

Overview

Hierarchical clustering

- Imagine we now want to create a hierachy in the form of a binary tree.

- Imagine we now want to create a hierachy in the form of a binary tree.
- Assumes a similarity measure for determining the similarity of two clusters.

- Imagine we now want to create a hierachy 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.

- Imagine we now want to create a hierachy 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 different cluster similarity measures.

Hierarchical clustering

- Imagine we now want to create a hierachy 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 different cluster similarity measures.
- Main algorithm: HAC (hierarchical agglomerative clustering)

- Start with each document in a separate cluster

- Start with each document in a separate cluster
- Then repeatedly merge the two clusters that are most similar

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.

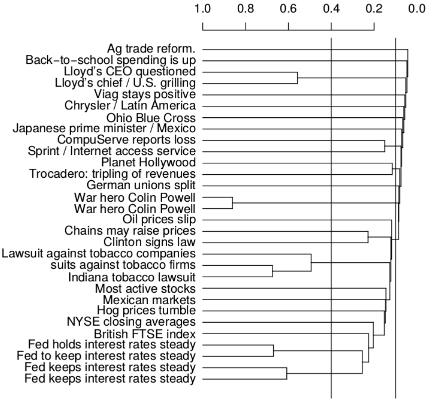
HAC: Basic algorithm

- 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.

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



▶ **Figure 17.1** A dendrogram of a single-link clustering of 30 documents from Reuters-RCV1. Two possible cuts of the dendrogram are shown: at 0.4 into 24 clusters and at 0.1 into 12 clusters.

Term–document matrix to document–document matrix

Log frequency weighting
and cosine normalisation

SaS	PaP	WH
0.789	0.832	0.524
0.515	0.555	0.465
0.335	0.000	0.405
0.000	0.000	0.588

SaS	P(SaS,SaS)	P(PaP,SaS)
PaP	P(SaS,PaP)	P(PaP,PaP)
WH	P(SaS,WH)	P(PaP,WH)
	SaS	PaP

SaS	1	.94	.79
PaP	.94	1	.69
WH	.79	.69	1
	SaS	PaP	WH

- Applying the proximity metric to all pairs of documents...
- creates the document-document matrix, which reports similarities/distances between objects (documents)

Term–document matrix to document–document matrix

Log frequency weighting and cosine normalisation

SaS	PaP	WH
0.789	0.832	0.524
0.515	0.555	0.465
0.335	0.000	0.405
0.000	0.000	0.588

SaS	P(SaS,SaS)	P(PaP,SaS)
PaP	P(SaS,PaP)	P(PaP,PaP)
WH	P(SaS,WH)	P(PaP,WH)
	SaS	PaP

SaS	1	.94	.79
PaP	.94	1	.69
WH	.79	.69	1
	SaS	PaP	WH

- Applying the proximity metric to all pairs of documents...
- creates the document-document matrix, which reports similarities/distances between objects (documents)

Term–document matrix to document–document matrix

Log frequency weighting and cosine normalisation

SaS	PaP	WH
0.789	0.832	0.524
0.515	0.555	0.465
0.335	0.000	0.405
0.000	0.000	0.588

SaS	P(SaS,SaS)	P(PaP,SaS)
PaP	P(SaS,PaP)	P(PaP,PaP)
WH	P(SaS,WH)	P(PaP,WH)
	SaS	PaP

SaS	1	.94	.79
PaP	.94	1	.69
WH	.79	.69	1
	SaS	PaP	WH

- Applying the proximity metric to all pairs of documents. . .
- creates the document-document matrix, which reports similarities/distances between objects (documents)
- The diagonal is trivial (identity)

Term–document matrix to document–document matrix

Log frequency weighting
and cosine normalisation

SaS	PaP	WH
0.789	0.832	0.524
0.515	0.555	0.465
0.335	0.000	0.405
0.000	0.000	0.588

SaS	P(SaS,SaS)	P(PaP,SaS)
PaP	P(SaS,PaP)	P(PaP,PaP)
WH	P(SaS,WH)	P(PaP,WH)
	SaS	PaP

SaS	1	.94	.79
PaP	.94	1	.69
WH	.79	.69	1
	SaS	PaP	WH

- Applying the proximity metric to all pairs of documents. . .
- creates the document-document matrix, which reports similarities/distances between objects (documents)

- As proximity measures are symmetric, the matrix is a triangle

Term–document matrix to document–document matrix

Log frequency weighting
and cosine normalisation

SaS	PaP	WH
0.789	0.832	0.524
0.515	0.555	0.465
0.335	0.000	0.405
0.000	0.000	0.588

SaS	P(SaS,SaS)	P(PaP,SaS)
PaP	P(SaS,PaP)	P(PaP,PaP)
WH	P(SaS,WH)	P(PaP,WH)
	SaS	PaP

SaS	1	.94	.79
PaP	.94	1	.69
WH	.79	.69	1
	SaS	PaP	WH

- Applying the proximity metric to all pairs of documents...
- creates the document-document matrix, which reports similarities/distances between objects (documents)

Term–document matrix to document–document matrix

Log frequency weighting and cosine normalisation

SaS	PaP	WH
0.789	0.832	0.524
0.515	0.555	0.465
0.335	0.000	0.405
0.000	0.000	0.588

SaS	P(SaS,SaS)	P(PaP,SaS)
PaP	P(SaS,PaP)	P(PaP,PaP)
WH	P(SaS,WH)	P(PaP,WH)
	SaS	PaP

SaS		.94	.79
PaP	.94		.69
WH	.79	.69	
	SaS	PaP	WH

- Applying the proximity metric to all pairs of documents...
- creates the document-document matrix, which reports similarities/distances between objects (documents)

Term–document matrix to document–document matrix

Log frequency weighting and cosine normalisation

SaS	PaP	WH
0.789	0.832	0.524
0.515	0.555	0.465
0.335	0.000	0.405
0.000	0.000	0.588

SaS	P(SaS,SaS)	P(PaP,SaS)
PaP	P(SaS,PaP)	P(PaP,PaP)
WH	P(SaS,WH)	P(PaP,WH)
	SaS	PaP

SaS		.94	.79
PaP	.94		.69
WH	.79	.69	
	SaS	PaP	WH

- Applying the proximity metric to all pairs of documents. . .
- creates the document-document matrix, which reports similarities/distances between objects (documents)

Hierarchical clustering: agglomerative (BottomUp, greedy)

Given: a set $X = x_1, ...x_n$ of objects;
Given: a function $sim : \mathcal{P}(X) \times \mathcal{P}(X) \rightarrow \mathcal{R}$

for i:= 1 **to** n **do**
    $c_i := x_i$
C $:= c_1$, ... c_n
j := n+1
while C $>$ 1 **do**
    $(c_{n_1}, c_{n_2}) := max_{(c_u, c_v) \in C \times C} sim(c_u, c_v)$
    $c_j := c_{n_1} \cup c_{n_2}$
    C := C $\{ c_{n_1}, c_{n_2} \} \cup c_j$
    j:=j+1
end

Similarity function $sim : \mathcal{P}(X) \times \mathcal{P}(X) \rightarrow \mathcal{R}$ measures similarity between clusters, not objects

Computational complexity of the basic algorithm

- First, we compute the similarity of all $N \times N$ pairs of documents.

Computational complexity of the basic algorithm

- First, we compute the similarity of all $N \times N$ pairs of documents.
- Then, in each of N iterations:

Computational complexity of the basic 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.

- 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.

Computational complexity of the basic 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.

Computational complexity of the basic 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.

Computational complexity of the basic 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 $O(N^3)$.

Computational complexity of the basic algorithm

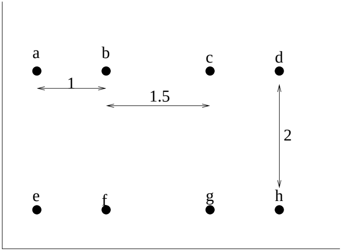
- First, we compute the similarity of all $N \times N$ pairs of documents.
- Then, in each of N iterations:
  - We scan the $O(N \times N)$ similarities to find the maximum similarity.
  - We merge the two clusters with maximum similarity.
  - We compute the similarity of the new cluster with all other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

Hierarchical clustering: similarity functions

Similarity between two clusters c_k and c_j (with similarity measure s) can be interpreted in different ways:

- Single Link Function: Similarity of two most similar members
$$sim(c_u, c_v) = max_{x \in c_u, y \in c_k} s(x, y)$$
- Complete Link Function: Similarity of two least similar members
$$sim(c_u, c_v) = min_{x \in c_u, y \in c_k} s(x, y)$$
- Group Average Function: Avg. similarity of each pair of group members
$$sim(c_u, c_v) = avg_{x \in c_u, y \in c_k} s(x, y)$$

Cluster 8 objects a-h; Euclidean distances (2D) shown in diagram



b	1						
c	2.5	1.5					
d	3.5	2.5	1				
e	2	$\sqrt{5}$	$\sqrt{10.25}$	$\sqrt{16.25}$			
f	$\sqrt{5}$	2	$\sqrt{6.25}$	$\sqrt{10.25}$	1		
g	$\sqrt{10.25}$	$\sqrt{6.25}$	2	$\sqrt{5}$	2.5	1.5	
h	$\sqrt{16.25}$	$\sqrt{10.25}$	$\sqrt{5}$	2	3.5	2.5	1
	a	b	c	d	e	f	g

Single Link is $O(n^2)$

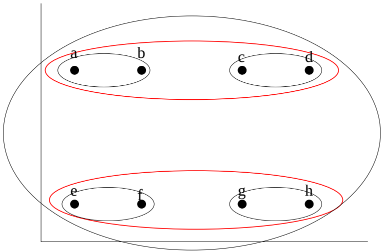
b	1						
c	2.5	1.5					
d	3.5	2.5	1				
e	2	$\sqrt{5}$	$\sqrt{10.25}$	$\sqrt{16.25}$			
f	$\sqrt{5}$	2	$\sqrt{6.25}$	$\sqrt{10.25}$	1		
g	$\sqrt{10.25}$	$\sqrt{6.25}$	2	$\sqrt{5}$	2.5	1.5	
h	$\sqrt{16.25}$	$\sqrt{10.25}$	$\sqrt{5}$	2	3.5	2.5	1
	a	b	c	d	e	f	g

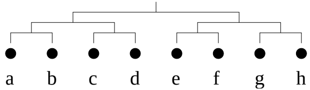
After Step 4 (a–b, c–d, e–f, g–h merged):

c–d	1.5		
e–f	2	$\sqrt{6.25}$	
g–h	$\sqrt{6.25}$	2	1.5
	a–b	c–d	e–f

"min-min" at each step

Clustering Result under Single Link





Complete Link

b	1						
c	2.5	1.5					
d	3.5	2.5	1				
e	2	$\sqrt{5}$	$\sqrt{10.25}$	$\sqrt{16.25}$			
f	$\sqrt{5}$	2	$\sqrt{6.25}$	$\sqrt{10.25}$	1		
g	$\sqrt{10.25}$	$\sqrt{6.25}$	2	$\sqrt{5}$	2.5	1.5	
h	$\sqrt{16.25}$	$\sqrt{10.25}$	$\sqrt{5}$	2	3.5	2.5	1
	a	b	c	d	e	f	g

After step 4 (a–b, c–d, e–f, g–h merged):

c–d	2.5	1.5				
	$\boxed{3.5}$	2.5				
e–f	2	$\sqrt{5}$	$\sqrt{10.25}$	$\boxed{\sqrt{16.25}}$		
	$\boxed{\sqrt{5}}$	2	$\sqrt{6.25}$	$\sqrt{10.25}$		
g–h	$\sqrt{10.25}$	$\sqrt{6.25}$	2	$\boxed{\sqrt{5}}$	2.5	1.5
	$\boxed{\sqrt{16.25}}$	$\sqrt{10.25}$	$\sqrt{5}$	2	$\boxed{3.5}$	2.5
	a–b		c–d		e–f	

"max-min" at each step

Complete Link

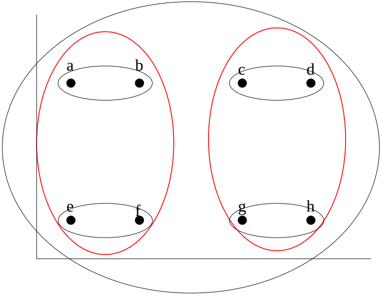
	a	b	c	d	e	f	g
b	1						
c	2.5	1.5					
d	3.5	2.5	1				
e	2	$\sqrt{5}$	$\sqrt{10.25}$	$\sqrt{16.25}$			
f	$\sqrt{5}$	2	$\sqrt{6.25}$	$\sqrt{10.25}$	1		
g	$\sqrt{10.25}$	$\sqrt{6.25}$	2	$\sqrt{5}$	2.5	1.5	
h	$\sqrt{16.25}$	$\sqrt{10.25}$	$\sqrt{5}$	2	3.5	2.5	1

After step 4 (a–b, c–d, e–f, g–h merged):

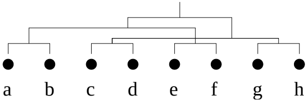
	a–b	c–d	e–f
c–d	2.5 1.5 $\boxed{3.5}$ 2.5		
e–f	2 $\sqrt{5}$ $\boxed{\sqrt{5}}$ 2	$\sqrt{10.25}$ $\boxed{\sqrt{16.25}}$ $\sqrt{6.25}$ $\sqrt{10.25}$	
g–h	$\sqrt{10.25}$ $\sqrt{6.25}$ $\boxed{\sqrt{16.25}}$ $\sqrt{10.25}$	2 $\boxed{\sqrt{5}}$ $\sqrt{5}$ 2	2.5 1.5 $\boxed{3.5}$ 2.5

"max-min" at each step → ab/ef and cd/gh merges next

Clustering result under complete link



Complete Link is $O(n^3)$



Example: gene expression data

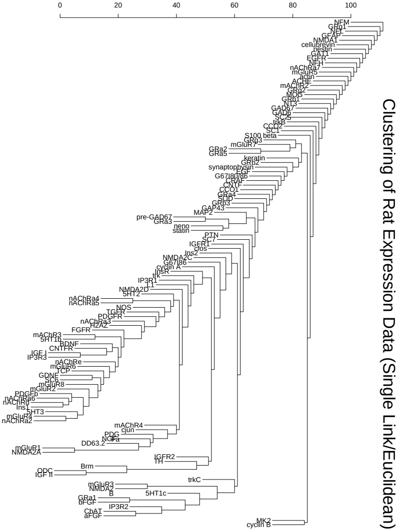
- An example from biology: cluster genes by function
- Survey 112 rat genes which are suspected to participate in development of CNS
- Take 9 data points: 5 embryonic (E11, E13, E15, E18, E21), 3 postnatal (P0, P7, P14) and one adult
- Measure expression of gene (how much mRNA in cell?)
- These measures are normalised logs; for our purposes, we can consider them as weights
- Cluster analysis determines which genes operate at the same time

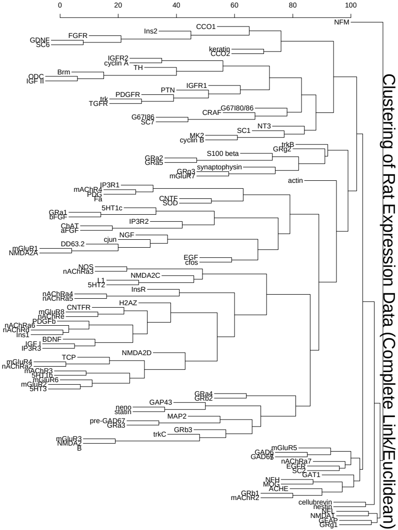
Rat CNS gene expression data (excerpt)

gene	genbank locus	E11	E13	E15	E18	E21	P0	P7	P14	A
keratin	RNKER19	1.703	0.349	0.523	0.408	0.683	0.461	0.32	0.081	0
cellubrevin	s63830	5.759	4.41	1.195	2.134	2.306	2.539	3.892	3.953	2.72
nestin	RATNESTIN	2.537	3.279	5.202	2.807	1.5	1.12	0.532	0.514	0.443
MAP2	RATMAP2	0.04	0.514	1.553	1.654	1.66	1.491	1.436	1.585	1.894
GAP43	RATGAP43	0.874	1.494	1.677	1.937	2.322	2.296	1.86	1.873	2.396
L1	S55536	0.062	0.162	0.51	0.929	0.966	0.867	0.493	0.401	0.384
NFL	RATNFL	0.485	5.598	6.717	9.843	9.78	13.466	14.921	7.862	4.484
NFM	RATNFM	0.571	3.373	5.155	4.092	4.542	7.03	6.682	13.591	27.692
NFH	RATNFHPEP	0.166	0.141	0.545	1.141	1.553	1.667	1.929	4.058	3.859
synaptophysin	RNSYN	0.205	0.636	1.571	1.476	1.948	2.005	2.381	2.191	1.757
neno	RATENONS	0.27	0.704	1.419	1.469	1.861	1.556	1.639	1.586	1.512
S100 beta	RATS100B	0.052	0.011	0.491	1.303	1.487	1.357	1.438	2.275	2.169
GFAP	RNU03700	0	0	0	0.292	2.705	3.731	8.705	7.453	6.547
MOG	RATMOG	0	0	0	0	0.012	0.385	1.462	2.08	1.816
GAD65	RATGAD65	0.353	1.117	2.539	3.808	3.212	2.792	2.671	2.327	2.351
pre-GAD67	RATGAD67	0.073	0.18	1.171	1.436	1.443	1.383	1.164	1.003	0.985
GAD67	RATGAD67	0.297	0.307	1.066	2.796	3.572	3.182	2.604	2.307	2.079
G67I80/86	RATGAD67	0.767	1.38	2.35	1.88	1.332	1.002	0.668	0.567	0.304
G67I86	RATGAD67	0.071	0.204	0.641	0.764	0.406	0.202	0.052	0.022	0
GAT1	RATGABAT	0.839	1.071	5.687	3.864	4.786	4.701	4.879	4.601	4.679
ChAT	(*)	0	0.022	0.369	0.322	0.663	0.597	0.795	1.015	1.424
ACHE	S50879	0.174	0.425	1.63	2.724	3.279	3.519	4.21	3.885	3.95
ODC	RATODC	1.843	2.003	1.803	1.618	1.569	1.565	1.394	1.314	1.11
TH	RATTOHA	0.633	1.225	1.007	0.801	0.654	0.691	0.23	0.287	0
NOS	RRBNOS	0.051	0.141	0.675	0.63	0.86	0.926	0.792	0.646	0.448
GRa1	(#)	0.454	0.626	0.802	0.972	1.021	1.182	1.297	1.469	1.511

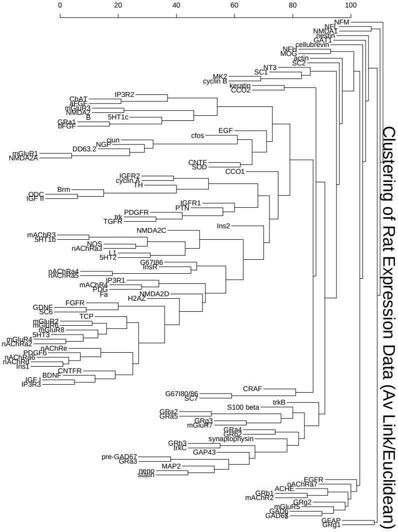
...

Rat CNS gene clustering – single link




Clustering of Rat Expression Data (Complete Link/Euclidean)
0
20
40
60
80
100
NFM
CCO1
Ins2
FGFR
GDNF
SC6
keratin
CCO2
IGFR2
cyclin A
TH
Brm
ODC
IGF II
IGFR1
PTN
PDGFR
trk
TGFR
G67I80/86
CRAF
G67I86
SC7
NT3
SC1
MK2
cyclin B
trkB
GRg2
S100 beta
GRa2
GRa5
synaptophysin
GRg3
mGluR7
actin
IP3R1
mAChR4
PDG
Fa
CNTF
SOD
5HT1c
GRa1
bFGF
IP3R2
ChAT
aFGF
NGF
cjun
DD63.2
mGluR1
NMDA2A
EGF
cfos
NOS
nAChRa3
NMDA2C
L1
5HT2
InsR
nAChRa4
nAChRa5
H2AZ
CNTFR
mGluR8
nAChRe
PDGFb
nAChRa6
nAChRd
Ins1
BDNF
IGF I
IP3R3
NMDA2D
TCP
mGluR4
nAChRa2
mAChR3
5HT1b
mGluR6
mGluR2
5HT3
GRa4
GRb2
GAP43
neno
statin
MAP2
pre-GAD67
GRa3
GRb3
trkC
mGluR3
NMDA2
B
mGluR5
GAD6
GAD65
nAChRa7
EGFR
SC2
GAT1
NFH
MOG
ACHE
GRb1
mAChR2
cellubrevin
nestin
NFL
NMDA1
GFAP
GRg1

Rat CNS gene clustering – group average link



- When a hierarchical structure is desired: hierarchical algorithm

Flat or hierarchical clustering?

- When a hierarchical structure is desired: hierarchical algorithm
- Humans are bad at interpreting hiearchical clusterings (unless cleverly visualised)

Flat or hierarchical clustering?

- When a hierarchical structure is desired: hierarchical algorithm
- Humans are bad at interpreting hiearchical clusterings (unless cleverly visualised)
- For high efficiency, use flat clustering

Flat or hierarchical clustering?

- When a hierarchical structure is desired: hierarchical algorithm
- Humans are bad at interpreting hiearchical clusterings (unless cleverly visualised)
- For high efficiency, use flat clustering
- For deterministic results, use HAC

Flat or hierarchical clustering?

- When a hierarchical structure is desired: hierarchical algorithm
- Humans are bad at interpreting hiearchical clusterings (unless cleverly visualised)
- For high efficiency, use flat clustering
- For deterministic results, use HAC
- HAC also can be applied if K cannot be predetermined (can start without knowing K)

Take-away

- Partitional clustering
  - Provides less information but is more efficient (best: $O(kn)$)
  - K-means
- Hierarchical clustering
  - Best algorithms $O(n^2)$ complexity
  - Single-link vs. complete-link (vs. group-average)
- Hierarchical and non-hierarchical clustering fulfills different needs

Reading

- MRS Chapters 16.1-16.4
- MRS Chapters 17.1-17.2