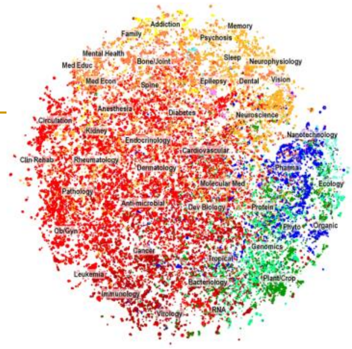
Unsupervised Learning
Text Clustering

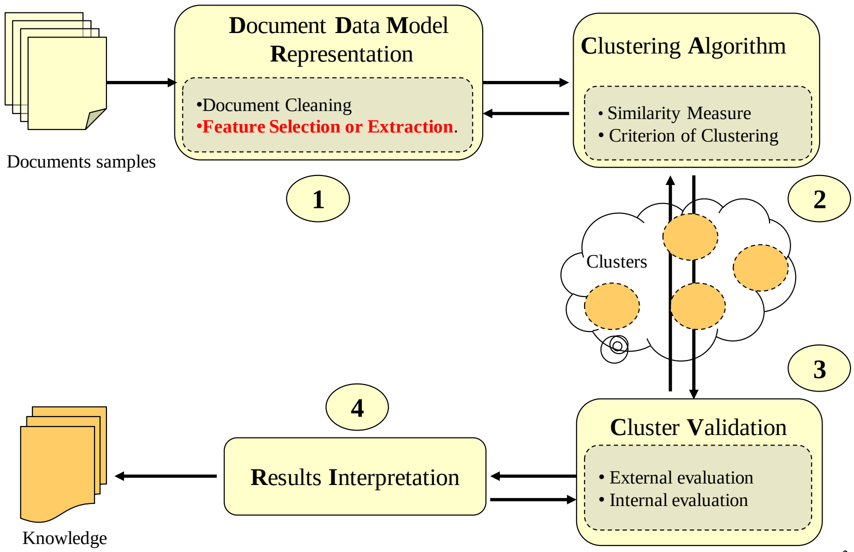


Prof. Rinaldo Lima – PPGIA - UFRPE

Topics

- **Basic concepts**
  - **Data Types and Representation**
  - **Vector Space Model**
  - **Distance Measures**
- **Unsupervised Clustering**
- **K-means algorithm**
  - **Representation of clusters**
- **Hierarchical clustering**
- **Which clustering algorithm to use?**
- **Cluster evaluation**
- **Summary**

Clustering Process



Vector Space Model

Text Representation (**Binary**)

Document 1

The Army needs senor technology to help find improvised explosive devices

Terms

Army
Sensor
Technology
Help
Find
Improvise
Explosive
device

Document 2

ORNL has developed sensor technology for homeland defense

ORNL
develop
sensor
technology
homeland
defense

Document 3

Mitre has won a contract to develop homeland defense sensors for explosive devices

Mitre
won
contract
develop
homeland
defense
sensor
explosive
devices

Term List

Army
Sensor
Technology
Help
Find
Improvise
Explosive
Device
ORNL
develop
homeland
Defense
Mitre
won
contract

Vector Space Model

	Doc 1	Doc 2	Doc 3
Army	1	0	0
Sensor	1	1	1
Technology	1	1	0
Help	1	0	0
Find	1	0	0
Improvise	1	0	0
Explosive	1	0	1
Device	1	0	1
ORNL	0	1	0
develop	0	1	1
homeland	0	1	1
Defense	0	1	1
Mitre	0	0	1
won	0	0	1
contract	0	0	1

The Vector Space Model

- Assume *t* distinct terms remain after preprocessing; call them index terms or the vocabulary.
- These "orthogonal" terms form a vector space.

$$\text{Dimension} = t = |\text{vocabulary}|$$

- Each term, *i*, in a document or query, *j*, is given a real-valued weight, w_{ij}.
- Both documents and queries are expressed as *t*-dimensional vectors:

$$d_j = (w_{1j}, w_{2j}, \ldots, w_{tj})$$

- New document is assigned to the most likely category based on vector similarity.

Document Collection

- A collection of n documents can be represented in the vector space model by a term-document matrix.
- An entry in the matrix corresponds to the *"weight" of a term in the document;* zero means the term has no significance in the document or it simply doesn't exist in the document.

$$
\begin{array}{c}
 \\ \mathbf{D_1} \\ \mathbf{D_2} \\ \vdots \\ \vdots \\ \mathbf{D_n}
\end{array}
\begin{pmatrix}
\mathbf{T_1} & \mathbf{T_2} & \mathbf{....} & \mathbf{T_t} \\
w_{11} & w_{21} & \cdots & w_{t1} \\
w_{12} & w_{22} & \cdots & w_{t2} \\
\vdots & \vdots & & \vdots \\
\vdots & \vdots & & \vdots \\
w_{1n} & w_{2n} & \cdots & w_{tn}
\end{pmatrix}
$$

Term Weights: Term Frequency

- More frequent terms in a document are more important, i.e. more indicative of the topic.

f_{ij} = frequency of term i in document j

- May want to normalize *term frequency* (*tf*) by dividing by the frequency of the most common term in the document:

$$tf_{ij} = f_{ij} / max_i\{f_{ij}\}$$

Term Weights: Inverse Document Frequency

- Terms that appear in many *different* documents are *less* indicative of overall topic

df_i = document frequency of term i
= number of documents containing term i
idf_i = inverse document frequency of term i,
$= \log_2 (N/ df_i)$
(N: total number of documents)

- An indication of a term's *discrimination* power.
- Log used to dampen the effect relative to *tf*.

TF-IDF Weighting

- A typical combined term importance indicator is *tf-idf weighting*:

$$w_{ij} = tf_{ij}\, idf_i = tf_{ij} \log_2 (N/ df_i)$$

- A term occurring frequently in the document but rarely in the rest of the collection is given high weight.
- Many other ways of determining term weights have been proposed.
- Experimentally, *tf-idf* has been found to work well.

TF-IDF Weighting : Example

- Given a document containing terms with given frequencies:

  A(3), B(2), C(1)

- Assume collection contains 10,000 documents and

  document frequencies of these terms are:

  A(50), B(1300), C(250)

$$tf_{ij}\, idf_i = tf_{ij} \log_2 (N/\, df_i)$$

Then:

normalized

A: $\boxed{tf = 3/3;}$ $idf = \log_2(10000/50) = 7.6;$ $\boxed{\text{tf-idf} = 7.6}$

B: $\boxed{tf = 2/3;}$ $idf = \log_2(10000/1300) = 2.9;$ $\boxed{\text{tf-idf} = 2.0}$

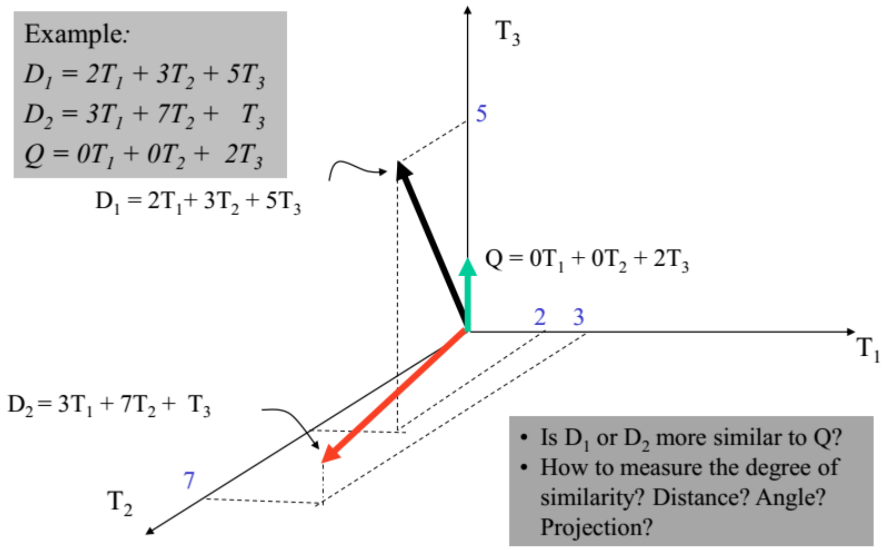
C: $\boxed{tf = 1/3;}$ $idf = \log_2(10000/250) = 5.3;$ $\boxed{\text{tf-idf} = 1.8}$

Similarity Measure

- A similarity measure is a function that computes the *degree of similarity* between two vectors.
- Using a similarity measure between the query and each document:
  - It is possible to rank the retrieved documents in the order of presumed relevance.
  - It is possible to enforce a certain threshold so that the size of the retrieved set can be controlled.

Geometric Interpretation

Example:
$D_1 = 2T_1 + 3T_2 + 5T_3$
$D_2 = 3T_1 + 7T_2 + T_3$
$Q = 0T_1 + 0T_2 + 2T_3$

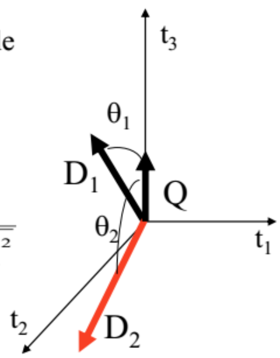


- Is D_1 or D_2 more similar to Q?
- How to measure the degree of similarity? Distance? Angle? Projection?

Similarity Measure: Cosine

- Cosine similarity measures the cosine of the angle between two vectors.
- Inner product normalized by the vector lengths.

$$\text{CosSim}(\mathbf{d}_j, \mathbf{q}) = \frac{\vec{d_j} \cdot \vec{q}}{|\vec{d_j}| \cdot |\vec{q}|} = \frac{\sum_{i=1}^{t}(w_{ij} \cdot w_{iq})}{\sqrt{\sum_{i=1}^{t} w_{ij}^2 \cdot \sum_{i=1}^{t} w_{iq}^2}}$$



$D_1 = 2T_1 + 3T_2 + 5T_3$ CosSim(D_1 , Q) = 10 / $\sqrt{(4+9+25)(0+0+4)}$ = 0.81
$D_2 = 3T_1 + 7T_2 + 1T_3$ CosSim(D_2 , Q) = 2 / $\sqrt{(9+49+1)(0+0+4)}$ = 0.13
$Q = 0T_1 + 0T_2 + 2T_3$

Distance Measures

Distance functions

- Key to clustering.
- "similarity" and "dissimilarity" are commonly used terms

- There are numerous distance functions for
  - Different types of data
    - Numeric data
    - Nominal data
  - Different specific applications

Distance function for text documents

- A text document consists of a sequence of sentences and each sentence consists of a sequence of words.
- To simplify: a document is usually considered a "bag" of words in document clustering.
  - Sequence and position of words are ignored.
- A document is represented with a vector just like a normal data point.
- It is common to use similarity to compare two documents rather than distance.
  - The most commonly used similarity function is the **cosine similarity**.

Distance Measures – Numerical Attributes

Minkowski Distance (http://en.wikipedia.org/wiki/Minkowski_distance)

For $\mathbf{x} = (x_1\, x_2 \cdots x_n)$ and $\mathbf{y} = (y_1\, y_2 \cdots y_n)$

$$d(\mathbf{x},\mathbf{y}) = \left(|x_1 - y_1|^p + |x_2 - y_2|^p \cdots + |x_n - y_n|^p\right)^{\frac{1}{p}}, \quad p > 0$$

– $p = 1$: Manhattan (city block) distance

$$d(\mathbf{x},\mathbf{y}) = |x_1 - y_1| + |x_2 - y_2| \cdots + |x_n - y_n|$$

– $p = 2$: Euclidean distance

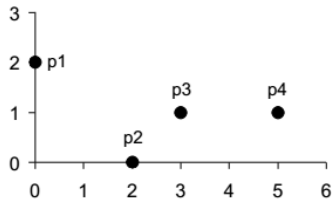
$$d(\mathbf{x},\mathbf{y}) = \sqrt{|x_1 - y_1|^2 + |x_2 - y_2|^2 \cdots + |x_n - y_n|^2}$$

– Do not confuse p with n, i.e., all these distances are defined based on all numbers of features (dimensions).

– A generic measure: use appropriate p in different applications

Distance Measures – Numerical Attributes

Example: Manhatten and Euclidean distances



L1	p1	p2	p3	p4
p1	0	4	4	6
p2	4	0	2	4
p3	4	2	0	2
p4	6	4	2	0

Distance Matrix for Manhattan Distance

point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1

Data Matrix

L2	p1	p2	p3	p4
p1	0	2.828	3.162	5.099
p2	2.828	0	1.414	3.162
p3	3.162	1.414	0	2
p4	5.099	3.162	2	0

Distance Matrix for Euclidean Distance

Distance Measures – Numerical Attributes

Cosine Measure (Similarity vs. Distance)

For $\mathbf{x} = (x_1\ x_2 \cdots x_n)$ and $\mathbf{y} = (y_1\ y_2 \cdots y_n)$

$$\cos(\mathbf{x}, \mathbf{y}) = \frac{x_1 y_1 + \cdots + x_n y_n}{\sqrt{x_1^2 + \cdots + x_n^2}\sqrt{y_1^2 + \cdots + y_n^2}}$$

$$d(\mathbf{x}, \mathbf{y}) = 1 - |\cos(\mathbf{x}, \mathbf{y})|$$

Or more general, **sim (x, y) = 1 – d(x, y)**

- Property: $0 \le d(\mathbf{x}, \mathbf{y}) \le 1$
- Nonmetric vector objects: keywords in documents, gene features in micro-arrays, …
- Applications: information retrieval, biologic taxonomy, ...

Distance Measures – Numerical Attributes

Example: Cosine measure

$$\mathbf{x}_1 = (3, 2, 0, 5, 2, 0, 0), \mathbf{x}_2 = (1,0, 0, 0, 1, 0, 2)$$

$$3\times1+2\times0+0\times0+5\times0+2\times1+0\times0+0\times2=5$$

$$\sqrt{3^2+2^2+0^2+5^2+2^2+0^2+0^2} = \sqrt{42} \approx 6.48$$

$$\sqrt{1^2+0^2+0^2+0^2+1^2+0^2+2^2} = \sqrt{6} \approx 2.45$$

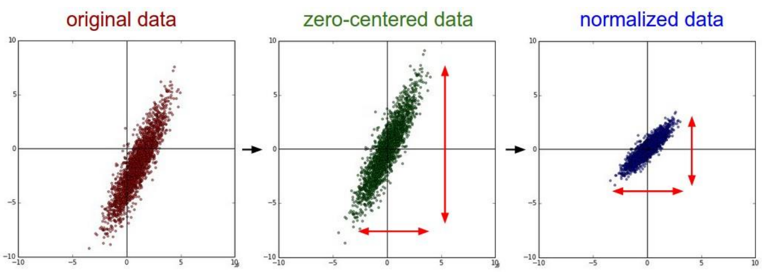
$$\cos(\mathbf{x}_1, \mathbf{x}_2) = \frac{5}{6.48\times2.45} \approx 0.32$$

$$d(\mathbf{x}_1, \mathbf{x}_2) = 1 - \cos(\mathbf{x}_1, \mathbf{x}_2) = 1 - 0.32 = 0.68$$

Data Normalization

- All of the above distance measures are sensible to difference in scales used to represent the measures
- A way to avoid such a problem is to normalize the data **before** representing and calculating the measure

Ex:



Data Normalization: Standardization vs Min-Max normalization

Min-Max Normalization: scales all the numeric values in the range [0..1] given by the formula:

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Standardization (Z-score): transform to a new scale in terms of unit variance (standard deviation) and mean = 0

$$x_{new} = \frac{x - \mu}{\sigma}$$

Distance functions for binary attributes

- Binary attribute: has two values or states but no ordering relationships, e.g.,
  - Gender: male and female.
- We use a **contigency matrix** to introduce the distance functions/measures.
- Let the *i*th and *j*th data points be a_i and b_j (vectors)

Distance Measures – Contigency Table

Distance for Binary Features

– For binary features, their value can be converted into 1 or 0.
– Contingency table for binary feature vectors, $\mathbf{x}$ and $\mathbf{y}$

		$\mathbf{y}$ 1	0
$\mathbf{x}$	1	a	b
	0	c	d

a : number of features that equal 1 for both $\mathbf{x}$ and $\mathbf{y}$

b : number of features that equal 1 for $\mathbf{x}$ but that are 0 for $\mathbf{y}$

c : number of features that equal 0 for $\mathbf{x}$ but that are 1 for $\mathbf{y}$

d : number of features that equal 0 for both $\mathbf{x}$ and $\mathbf{y}$

Symmetric binary attributes

- A binary attribute is **symmetric** if both of its states (0 and 1) have equal importance, and carry the same weights, e.g., male and female of the attribute Gender

- Distance function: Simple Mismatching Coefficient, proportion of mismatches of their values

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \frac{b + c}{a + b + c + d}$$

		$\mathbf{y}$ 1	0
$\mathbf{x}$	1	a	b
	0	c	d

Distance Measures

		y	
		1	0
x	1	a	b
	0	c	d

Distance for Binary Features

– Distance for symmetric binary features

Both of their states equally valuable and carry the same weight; i.e., no preference on which outcome should be coded as 1 or 0 , e.g. gender

$$d(\mathbf{x}, \mathbf{y}) = \frac{b+c}{a+b+c+d}$$

$\mathbf{x}_1$	1	1	1	0	1	0	0
$\mathbf{x}_2$	0	1	1	0	0	1	0

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \frac{2+1}{2+2+1+2} = \frac{3}{7} = 0.429$$

27

Asymmetric binary attributes

- Asymmetric: if one of the states is more important or more valuable than the other.
  - By convention, state 1 represents the more important state, which is typically the rare or infrequent state.
  - Jaccard coefficient is a popular measure

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \frac{b+c}{a+b+c}$$

  - We can have some variations, adding weights

28

Distance Measures: **asymmetric**

		y	
		1	0
x	1	a	b
	0	c	d

Example: Distance for binary features

Name	Gender	Fever	Cough	Test-1	Test-2	Test-3	Test-4
Jack	M	1	0	1	0	0	0
Mary	F	1	0	1	0	1	0
Jim	M	1	1	0	0	0	0

- "Y": yes
- "P": positive
- "N": negative

- gender is a symmetric feature (less important)
- the remaining features are asymmetric binary
- set the values "Y" and "P" to 1, and the value "N" to 0

$$\begin{array}{cc} & \text{Mary} \\ \text{Jack} & \begin{array}{|c|c|}\hline 2 & 0 \\ \hline 1 & 3 \\ \hline\end{array}\end{array} \qquad d(\text{Jack},\text{Mary}) = \frac{0+1}{2+0+1} = 0.33$$

$$\begin{array}{cc} & \text{Jim} \\ \text{Jack} & \begin{array}{|c|c|}\hline 1 & 1 \\ \hline 1 & 3 \\ \hline\end{array}\end{array} \qquad d(\text{Jack},\text{Jim}) = \frac{1+1}{1+1+1} = 0.67$$

$$\begin{array}{cc} & \text{Mary} \\ \text{Jim} & \begin{array}{|c|c|}\hline 1 & 1 \\ \hline 2 & 2 \\ \hline\end{array}\end{array} \qquad d(\text{Jim},\text{Mary}) = \frac{1+2}{1+1+2} = 0.75$$

29

Distance Measure: Nominal attributes

A nominal attribute can have *n* states.

- Ex. Color: *yellow, white, black* (3 states)

The dissimilarity between two objects x and y can be computed based on the ratio of mismatches:

$$d(x,y) = \frac{p-m}{p} \qquad sim(x,y) = \frac{m}{p}$$

where, **m** is the number of mismatches and **p** is the total number of attributes

31

Nominal attributes to binary ones

- Transform nominal attributes to binary attributes.
  - The number of values of a nominal attribute is *v*.
  - Create *v* binary attributes to represent them.
  - If a data instance for the nominal attribute takes a particular value, the value of its binary attribute is set to 1, otherwise it is set to 0.
- The resulting binary attributes can be used as numeric attributes, with two values, 0 and 1.

Nominal attributes: an example

- Nominal attribute *fruit*: has three values,
  - Apple, Orange, and Pear
- We create three binary attributes called, Apple, Orange, and Pear in the new data.
- If a particular data instance in the original data has Apple as the value for *fruit*,
  - then in the transformed data, we set the value of the attribute Apple to 1, and
  - the values of attributes Orange and Pear to 0

Ordinal attributes

- Ordinal attribute: an ordinal attribute is like a nominal attribute, but its values have a numerical ordering.
- E.g.,
  - Age attribute with values: Young, MiddleAge and Old. They are ordered.
  - Common approach to standardization: treat is as an interval-scaled attribute.

Document Representation for Clustering

Document Representations

- Discrete vs. Continuous
  - Discrete Feature
    - Has only a finite set of values
      e.g., zip codes, rank, or the set of words in a collection of documents
    - Sometimes, represented as integer variable
  - Continuous Feature
    - Has real numbers as feature values
      e.g, temperature, height, or weight
    - Practically, real values can only be measured and represented using a finite number of digits
    - Continuous features are typically represented as floating-point variables

Document Representations

– Data matrix (object-by-feature structure)

$$\begin{bmatrix} x_{11} & \cdots & x_{1f} & \cdots & x_{1p} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{i1} & \cdots & x_{if} & \cdots & x_{ip} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{nf} & \cdots & x_{np} \end{bmatrix}$$

- **n data points (objects) with p dimensions (features)**
- **Two modes: row and column represent different entities**

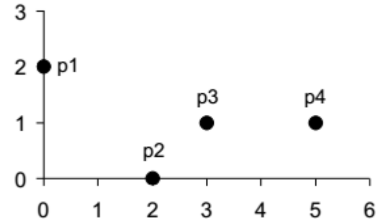
– Distance/dissimilarity matrix (object-by-object structure)

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & & \\ d(n,1) & d(n,2) & \cdots & \cdots & 0 \end{bmatrix}$$

- **n data points, but registers only the distance**
- **A symmetric/triangular matrix**
- **Single mode: row and column for the same entity (distance)**

Document Representations

Examples



point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1

Data Matrix

	p1	p2	p3	p4
p1	0	2.828	3.162	5.099
p2	2.828	0	1.414	3.162
p3	3.162	1.414	0	2
p4	5.099	3.162	2	0

Distance Matrix (i.e., Dissimilarity Matrix) for Euclidean Distance

Unsupervised Learning

Clustering

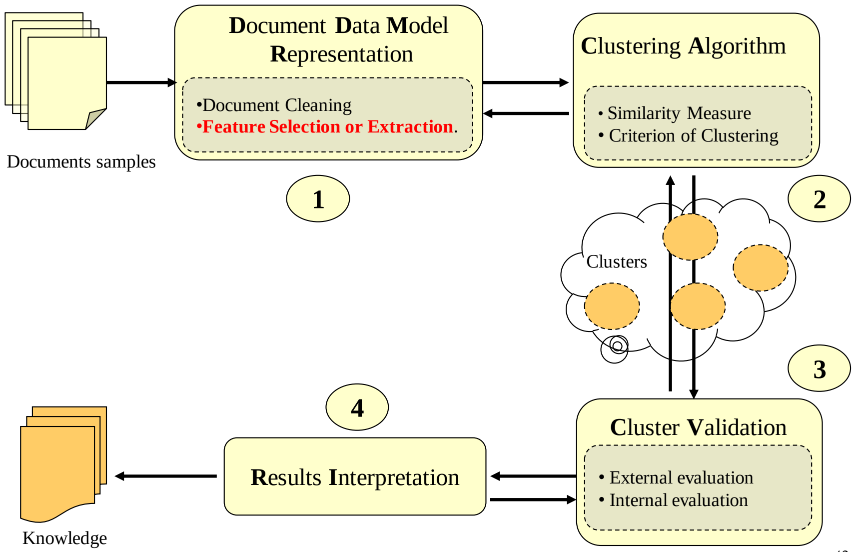
Supervised learning vs. unsupervised learning

- Supervised learning: discover patterns in the data that relate data attributes with a target (**class**) attribute.
  - These patterns are then utilized to predict the values of the target attribute in future data instances.
- Unsupervised learning: The data have no target attribute.
  - We want to explore the data to find some intrinsic structures in them.

Clustering

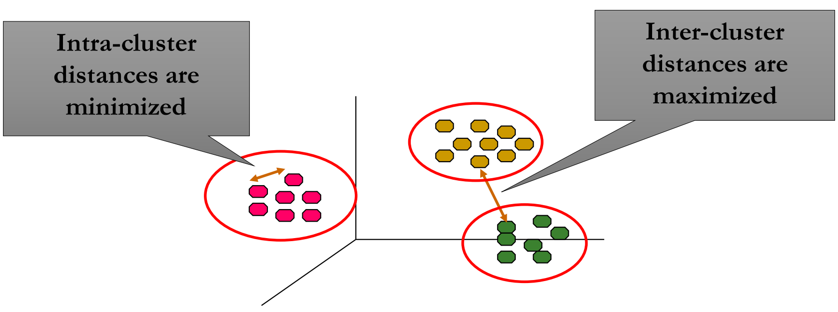
- Clustering is a technique for finding similarity groups in data, called **clusters**. I.e.,
  - it groups data instances that are similar to (near) each other in one cluster and data instances that are very different (far away) from each other into different clusters.
  - **Cluster Hypothesis** : Relevant documents tend to be more similar to each other than to non-relevant ones.
- Clustering is often called an **unsupervised learning** task as no class values denoting an *a priori* grouping of the data instances are given, which is the case in supervised learning.

Clustering Process



An illustration

- The data set has three natural groups of data points, i.e., 3 natural clusters.



What is clustering for?

Let us see some real-life examples

- Example 1: groups people of similar sizes together to make “small”, “medium” and “large” T-Shirts.

- Example 2: In marketing, segment customers according to their similarities
  - To do targeted marketing.

44

What is clustering for? (cont…)

- Example 3: Given a collection of text documents, we want to organize them according to their content similarities,
  - To produce a topic hierarchy
- In fact, clustering is one of the most utilized data mining techniques.
  - It has a long history, and used in almost every field, e.g., medicine, psychology, botany, sociology, biology, archeology, marketing, insurance, libraries, etc.
  - In recent years, due to the rapid increase of online documents, text clustering becomes important.

45

Aspects of clustering

- A clustering algorithm
  - **Partitional clustering**
  - **Hierarchical clustering**
  - Density-based clustering
  - Graph-based Clustering
- A distance (similarity, or dissimilarity) function
- Clustering quality
  - Inter-clusters distance $\Rightarrow$ maximized
  - Intra-clusters distance $\Rightarrow$ minimized
- The quality of a clustering result depends on the algorithm, the distance function, and the application.

Clustering

k-Means Algorithm

K-means clustering

- K-means is a partitional clustering algorithm
- Let the set of data points (or instances) D be

  $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$,

  where $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ir})$ is a vector in a real-valued space $X \subseteq R^r$, and r is the number of attributes (dimensions) in the data.
- The k-means algorithm partitions the given data into k clusters.
  - Each cluster has a cluster **center**, called **centroid**.
  - ***k*** is specified by the user

K-means algorithm

- Given k, the *k-means* algorithm works as follows:
  1) Randomly choose k data points (seeds) to be the initial centroids, **cluster centers**
  2) Assign each data point to the closest centroid
  3) Re-compute the centroids using the current cluster memberships.
  4) If a convergence criterion is not met, go to 2) else stop

K-means algorithm – (cont …)

Algorithm *k*-means(*k*, *D*)
Choose *k* data points as the initial centroids (cluster centers)
repeat
    for each data point $\mathbf{x} \in D$ do
        compute the distance from **x** to each centroid;
        assign **x** to the closest centroid   // a centroid represents a cluster
    endfor
    re-compute the centroids using the current cluster memberships
until the stopping criterion is met

Stopping/convergence criterion

1. no (or minimum) re-assignments of data points to different clusters,
2. no (or minimum) change of centroids, or
3. minimum decrease in the **sum of squared error** (SSE),

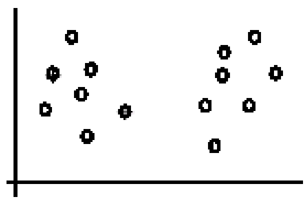
$$SSE = \sum_{j=1}^{k} \sum_{\mathbf{x} \in C_j} dist(\mathbf{x}, \mathbf{m}_j)^2 \qquad (1)$$

- C_j is the *j*th cluster, $\mathbf{m}_j$ is the centroid of cluster C_j (the mean vector of all the data points in C_j), and $dist(\mathbf{x}, \mathbf{m}_j)$ is the distance between data point **x** and centroid $\mathbf{m}_j$.

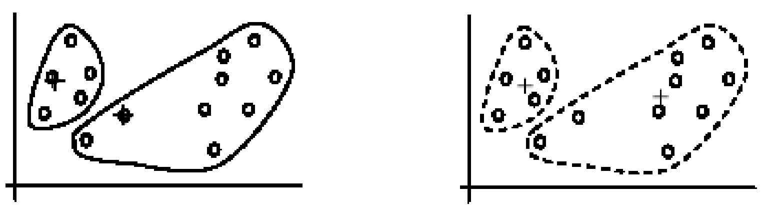
k-Means Algorithm

Example

An example



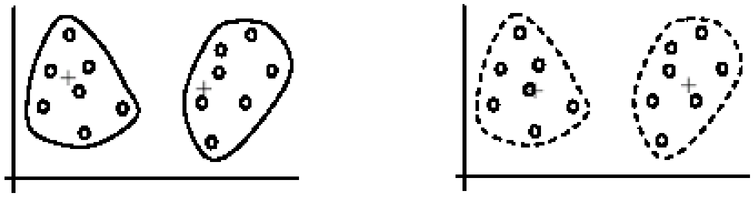
(A). Random selection of *k* centers



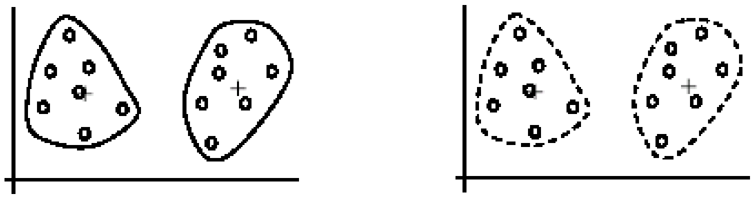
Iteration 1: (B). Cluster assignment

(C). Re-compute centroids

An example (cont …)

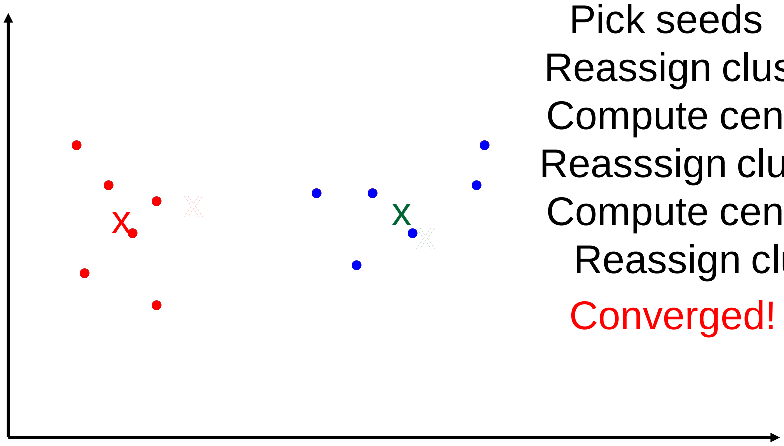


Iteration 2: (D). Cluster assignment (E). Re-compute centroids



Iteration 3: (F). Cluster assignment (G). Re-compute centroids

K Means Example (K=2)



Pick seeds
Reassign clusters
Compute centroids
Reasssign clusters
Compute centroids
Reassign clusters
Converged!

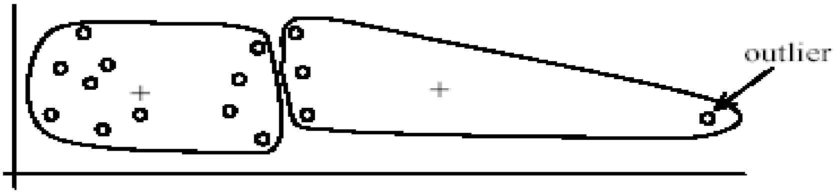
Strengths of k-means

- Strengths:
  - Simple: easy to understand and to implement
  - Efficient: Time complexity: $O(TKN)$, where
    - n is the number of data points,
    - k is the number of clusters, and
    - t is the number of iterations.
  - Since both k and t are small. k-means is considered a **linear algorithm.**
- K-means is the most popular clustering algorithm.
- It terminates at a local optimum if SSE is used.
- The global optimum is hard to find due to complexity.

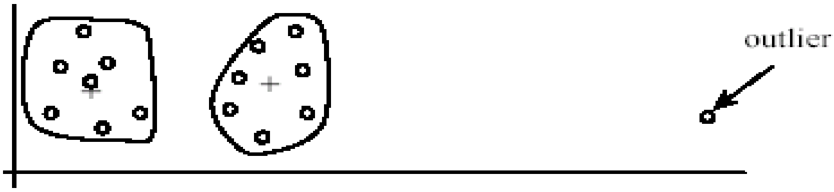
Weaknesses of k-means

- The algorithm is only applicable if the mean is defined.
- The user needs to specify k.
- The algorithm is sensitive to **outliers**
  - Outliers are data points that are very far away from other data points.
  - Outliers could be errors in the data recording or some special data points with very different values.

Weaknesses of k-means: Problems with outliers



(A): Undesirable clusters



(B): Ideal clusters

58

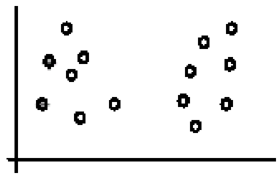
Weaknesses of k-means: dealing with outliers

- One method is to remove some data points in the clustering process that are much further away from the centroids than other data points.
  - To be safe, we may want to monitor these possible outliers over a few iterations and then decide to remove them.
- Another method is to perform **random sampling**. Since in sampling we only choose a small subset of the data points, the chance of selecting an outlier is very small.
  - Assign the rest of the data points to the clusters by distance or similarity comparison, or classification

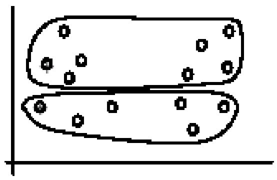
59

Weaknesses of k-means (cont …)

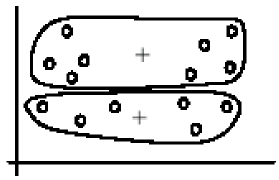
- The algorithm is sensitive to initial seeds.



(A). Random selection of seeds (centroids)



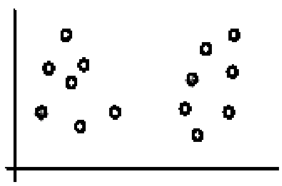
(B). Iteration 1



(C). Iteration 2

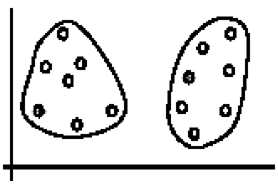
Weaknesses of k-means (cont …)

- If we use different seeds: good results

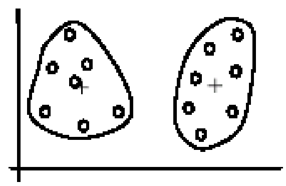


- There are many methods to help choose good seeds

(A). Random selection of k seeds (centroids)



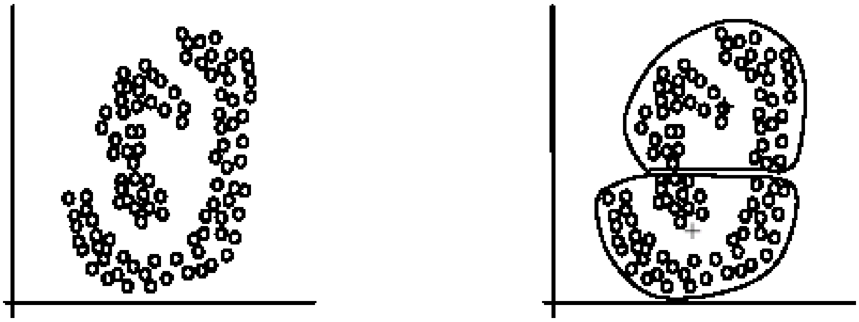
(B). Iteration 1



(C). Iteration 2

Weaknesses of k-means (cont …)

- The *k*-means algorithm is not suitable for discovering clusters that are **not hyper-ellipsoids** (or hyper-spheres).



(A): Two natural clusters (B): *k*-means clusters

K-means summary

- Despite weaknesses, *k*-means is still the most popular algorithm due to its simplicity, efficiency and
  - other clustering algorithms have their own lists of weaknesses.
- No clear evidence that any other clustering algorithm performs better in general
  - although they may be more suitable for some specific types of data or applications.
- Comparing different clustering algorithms is a difficult task.
- No one knows the correct clusters!

Clustering

Centroids

Common ways to represent clusters

- Use the centroid of each cluster to represent the cluster.
- **standard deviation** of the cluster to determine its spread in each dimension
- The centroid representation alone works well if the clusters are of the hyper-spherical shape.
- If clusters are elongated or are of other shapes, centroids are not sufficient

Definition of centroid

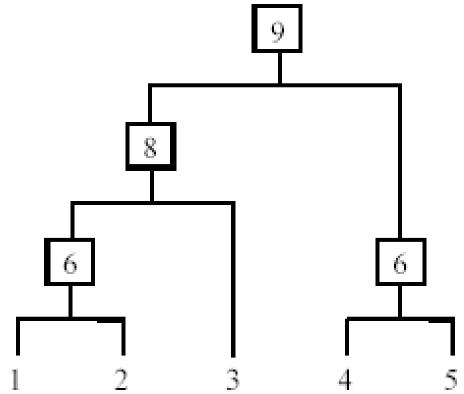
$$\vec{\mu}(c) = \frac{1}{|D_c|} \sum_{d \in D_c} \vec{v}(d)$$

- Where D_c is the set of all documents that belong to class c and $v(d)$ is the vector space representation of d.
- *Note that centroid will in general* not *be a unit vector even when the inputs are unit vectors*
- A well-known variant of k-means is the **k-medoids**: just use an actual example as centroid of a cluster

Hierarchical Agglomerative Clustering (HAC)

Hierarchical Clustering

- Produce a nested sequence of clusters, a tree, also called Dendrogram.



Types of hierarchical clustering

- Agglomerative (bottom up) clustering: It builds the dendrogram (tree) from the bottom level, and
  - merges the most similar (or nearest) pair of clusters
  - stops when all the data points are merged into a single cluster (i.e., the root cluster).
- Divisive (top down) clustering: It starts with all data points in one cluster, the root.
  - Splits the root into a set of child clusters. Each child cluster is recursively divided further
  - stops when only singleton clusters of individual data points remain, i.e., each cluster with only a single point

Agglomerative clustering

It is more popular then divisive methods.

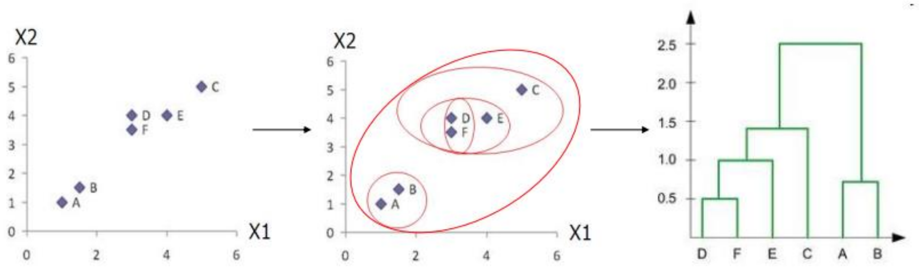
- At the beginning, each data point forms a cluster (also called a node).
- Merge nodes/clusters that have the least distance.
- Go on merging
- Eventually all nodes belong to one cluster

Agglomerative clustering algorithm

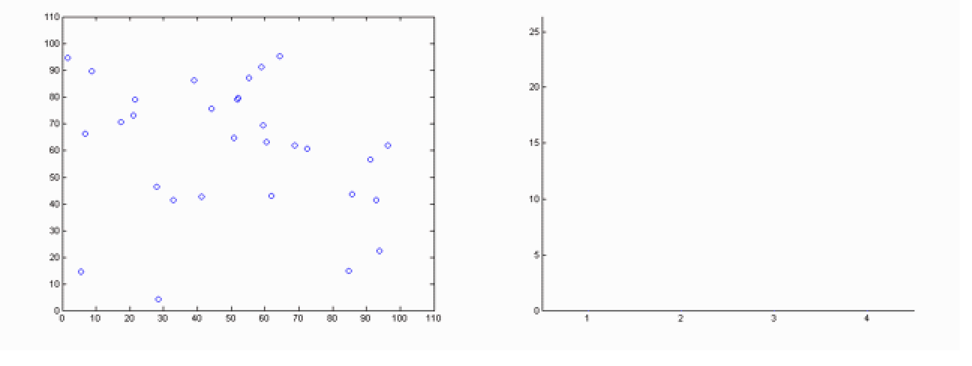
Algorithm Agglomerative(D)

Make each data point in the data set D a cluster,
Compute all pair-wise distances of $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in D$;
repeat
    find two clusters that are nearest to each other;
    merge the two clusters form a new cluster c;
    compute the distance from c to all other clusters;
until there is only one cluster left

Working example of the algorithm



Hierarchical Clustering Demo



Which Cluster Algorithm to use?

How to choose a clustering algorithm

- Clustering research has a long history. A vast collection of algorithms are available.
  - We only introduced the main algorithms.
- Choosing the "best" algorithm is challenging
  - Every algorithm has limitations and works well with certain data distributions.
  - It is very hard, if not impossible, to know what distribution the application data follow. The data may not fully follow any "ideal" structure or distribution required by the algorithms.
  - One also needs to decide how to standardize the data, to choose a suitable distance function and to select other parameter values.

Choose a clustering algorithm (cont …)

- Due to these complexities, the common practice is to
  - run several algorithms using different distance functions and parameter settings, and
  - then carefully analyze and compare the results.
- The interpretation of the results must be based on insight into the meaning of the original data together with knowledge of the algorithms used.
- Clustering is highly application dependent and to certain extent subjective (personal preferences).

Cluster Evaluation

Cluster Evaluation: hard problem

- The quality of a clustering is very hard to evaluate because
  - We do not know the correct clusters
- Some methods are used:
  - User inspection
    - Study **centroids**, and **spreads**
    - For text documents, one can read some documents in clusters.

Cluster evaluation: ground truth

- We use some labeled data (for classification)
- Assumption: Each class is a cluster.
- After clustering, a confusion matrix is constructed.
- From the matrix, we compute various measurements, including **precision, recall and F-score**.
  - Let the classes in the data D be $C = (c_1, c_2, \ldots, c_k)$. The clustering method produces k clusters, which divides D into k disjoint subsets, $D_1, D_2, \ldots, D_k$.

A remark about ground truth evaluation

- Commonly used to compare different clustering algorithms.
- A real-life data set for clustering has no class labels.
  - Thus although an algorithm may perform very well on some labeled data sets, no guarantee that it will perform well on the actual application data at hand.
- The fact that it performs well on some label data sets does give us some confidence of the quality of the algorithm.
- This evaluation method is said to be based on **external data** or information.

Squared Errors and Cluster Centers

- Squared error (distance) between a data point x and a cluster center c:

$$d [x , c] = \Sigma_j (x_j - c_j)^2$$

- Total squared error between a cluster center c(k) and all N_k points assigned to that cluster:

$$S_k = \Sigma_i d [x_i , c_k]$$

Distance is usually defined as Euclidean distance

- Total squared error summed across K clusters

$$SSE = \Sigma_k S_k$$

Evaluation based on internal information

- **Intra-cluster cohesion** (compactness):
  - Cohesion measures how near the data points in a cluster are to the cluster centroid.
  - Sum of squared error (SSE) is a commonly used measure.
- **Inter-cluster separation** (isolation):
  - Separation means that different cluster centroids should be far away from one another.
- In most applications, expert judgments are still the key.

Indirect evaluation

- In some applications, clustering is not the primary task, but used to help perform another task.
- We can use the performance on the primary task to compare clustering methods.
- For instance, in an application, the primary task is to provide **recommendations on book purchasing** to online shoppers.
  - If we can cluster books according to their features, we might be able to provide better recommendations.
  - We can evaluate different clustering algorithms based on how well they help with the recommendation task.
  - Here, we assume that the recommendation can be reliably evaluated.

Review: Standard Textual Clustering

Vector Space Model

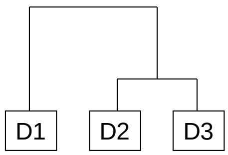
	Doc 1	Doc 2	Doc 3
Army	1	0	0
Sensor	1	1	1
Technology	1	1	0
Help	1	0	0
Find	1	0	0
Improvise	1	0	0
Explosive	1	0	1
Device	1	0	1
ORNL	0	1	0
develop	0	1	1
homeland	0	1	1
Defense	0	1	1
Mitre	0	0	1
won	0	0	1
contract	0	0	1

Dissimilarity Matrix

	Doc 1	Doc 2	Doc 3
Doc 1	100%	17%	21%
Doc 2		100%	36%
Doc 3			100%

Documents to Documents

Cluster Analysis



Most similar documents

TFIDF

$$W_{ij} = \log_2\left(f_{ij}+1\right)*\log_2\left(\frac{N}{n}\right)$$

Euclidean distance

$$d_2(\mathbf{x}_i, \mathbf{x}_j) = \left(\sum_{k=1}^{d}(x_{i,k} - x_{j,k})^2\right)^{1/2}$$

Summary

- Clustering is has along history and still active
  - There are a huge number of clustering algorithms
  - More are still coming every year.
- We only introduced several main algorithms. There are many others, e.g.,
  - density based algorithm, sub-space clustering, scale-up methods, neural networks based methods, fuzzy clustering, co-clustering, etc.
- Clustering is hard to evaluate, but very useful in practice.
- This partially explains why there are still a large number of clustering algorithms being devised every year.
  - Clustering is highly application dependent

Clustering Process