Clustering

(1IK172 Introduction to Data Analytics)

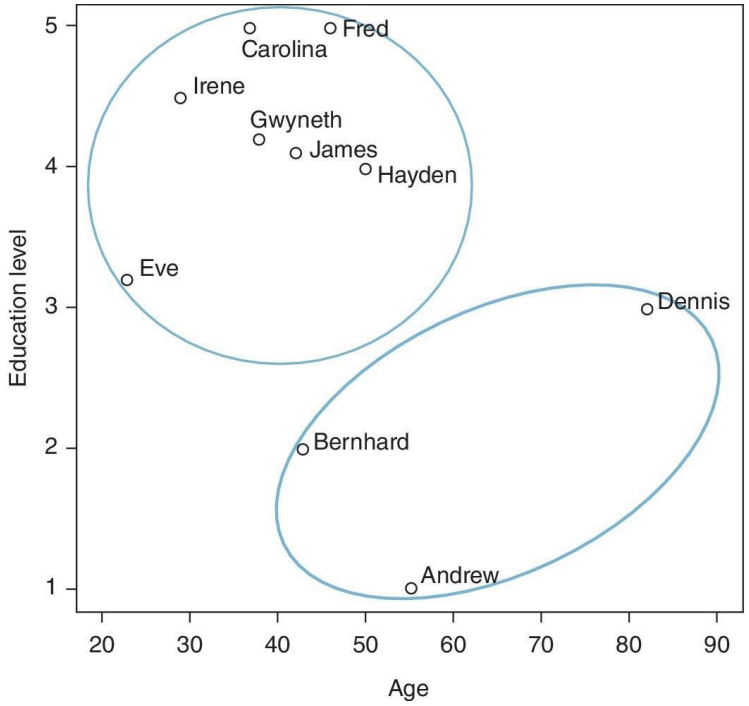
Problem definition

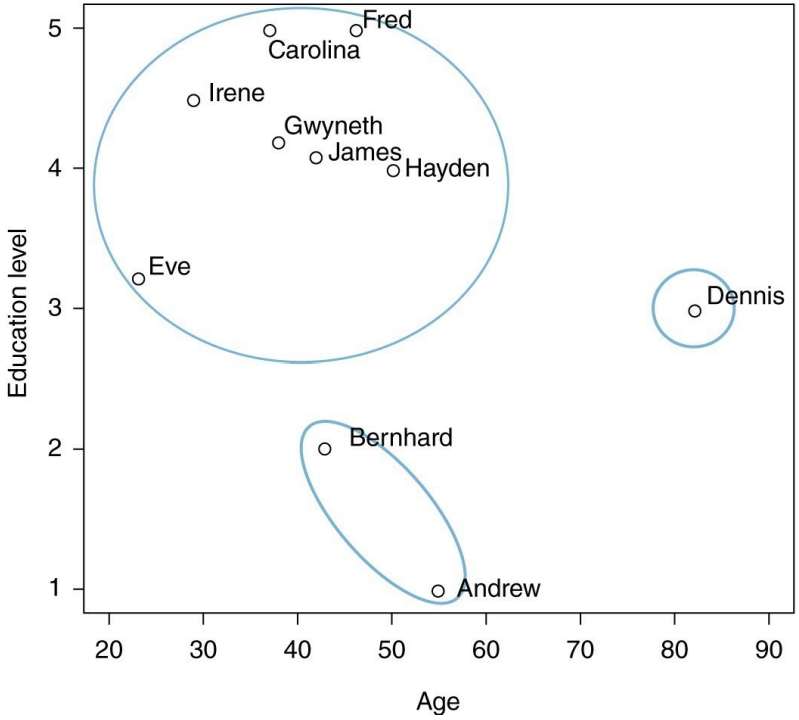
Partitioning the data

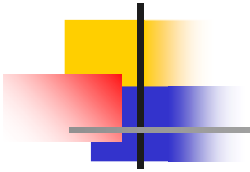
- Such that objects belonging to the same group (cluster) are more similar to each other than objects belonging to different groups
- Similarity of objects and groups is a key concept in clustering
- The number of clusters has to be set up by a human expert, in general

Name	Age	Educational level
Andrew	55	1
Bernhard	43	2
Carolina	37	5
Dennis	82	3
Eve	23	3.2
Fred	46	5
Gwyneth	38	4.2
Hayden	50	4
Irene	29	4.5
James	42	4.1

Example of clusters

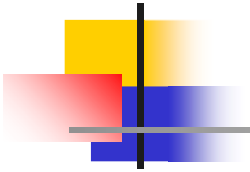






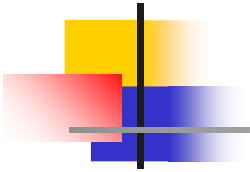
Problems with Clustering

- Clustering in two dimensions looks easy.
- Clustering small amounts of data looks easy.
- And in most cases, looks are *not* deceiving.



The Curse of Dimensionality

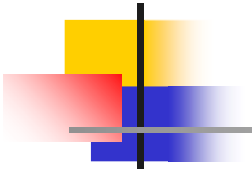
- Many applications involve not 2, but 10 or 10,000 dimensions.
- High-dimensional spaces look different: almost all pairs of points are at about the same distance.
  - Assuming random points within a bounding box, e.g., values between 0 and 1 in each dimension.



Example: SkyCat

- A catalog of 2 billion “sky objects” represented objects by their radiation in 9 dimensions (frequency bands).
- Problem: cluster into similar objects, e.g., galaxies, nearby stars, quasars, etc.

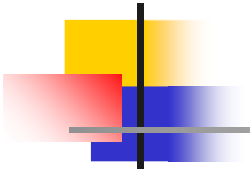
Example: Clustering CD's (Collaborative Filtering)



- Intuitively: music divides into categories, and customers prefer a few categories.
  - But what are categories really?
- Represent a CD by the customers who bought it.
- Similar CD's have similar sets of customers, and vice-versa.

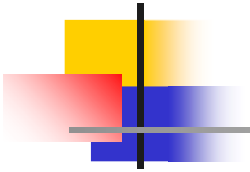
Example: Clustering Documents

- Represent a document by a vector $(x_1, x_2, \ldots, x_k)$, where $x_i = 1$ if the i^{th} word (in some order) appears in the document.
  - It actually doesn't matter if k is infinite; i.e., we don't limit the set of words.
- Documents with similar sets of words may be about the same topic.



Distance Measure

- Each clustering problem is based on some kind of "distance" between points.
- Two major classes of distance measure:
  1. ***Euclidean***
  2. ***Non-Euclidean***



Euclidean Vs. Non-Euclidean

- A *Euclidean space* has some number of real-valued dimensions and "dense" points.
  - There is a notion of "average" of two points.
  - A *Euclidean distance* is based on the locations of points in such a space.
- A *Non-Euclidean distance* is based on properties of points, but not their "location" in a space.

Similarity Measurement

- Quantitative attributes

$$d(a,b) = |a - b|$$

- Qualitative attributes (ordinal)

$$d(a,b) = \frac{|pos_a - pos_b|}{n-1}$$

  - Where n is the number of different values, and pos_a and pos_b are the positions of the values a and b, respectively, in a ranking of possible values.

- Qualitative attributes (nominal)

$$d(a,b) = \begin{cases} 1 & \text{if } a \neq b \\ 0 & \text{if } a = b \end{cases}$$

Example

Difference between the attributes of Andrew (a) and Carolina (b) are

- Age (Quantitative)
  - d(a,b) = d(55,37) = |55-37| = 18
- Education level (Ordinal)
  - d(a,b) = d(1,5) = $|pos_1 - pos_5|$ / 4 = |1-5| / 4 = 1
- Name (Nominal)
  - d(a,b) = d("Andrew","Carolina") = 1

Difference between sequences

Hamming distance

- Applicable for sequences of characters
- Calculates the number of positions at which the corresponding characters in the two strings are different
- Only applicable for strings of the same length

- **Example:**
  $d_{Hamming}$(“James”, “Jimmy”) = 3,
  $d_{Hamming}$(“Tom”, “Tim”) = 1

Difference between sequences

Levenshtein or edit distance

- Calculates the minimum number of operations necessary to transform one sequence into another
  - Insertion
  - Removal
  - Substitution
- Can be calculated for arbitrary strings
- Example
  d_{edit}("Johnny", "Jonston") = 5
  - 4 substitutions (h→n, n→s, n→t, y→o)
  - 1 insertion (adding "n" to the end)

Difference between sequences

Bag-of-words vector similarity

- Long texts converted to a quantitative vector where each position is associated with one of the words found and its value is the number of times this word appeared in the text
- Difference of two texts is calculated as the distance between their bag-of-word vectors
- **Example:**
  text A = "I will go to the party. But first, I will have to work."
  text B = "They have to go to the work by bus."

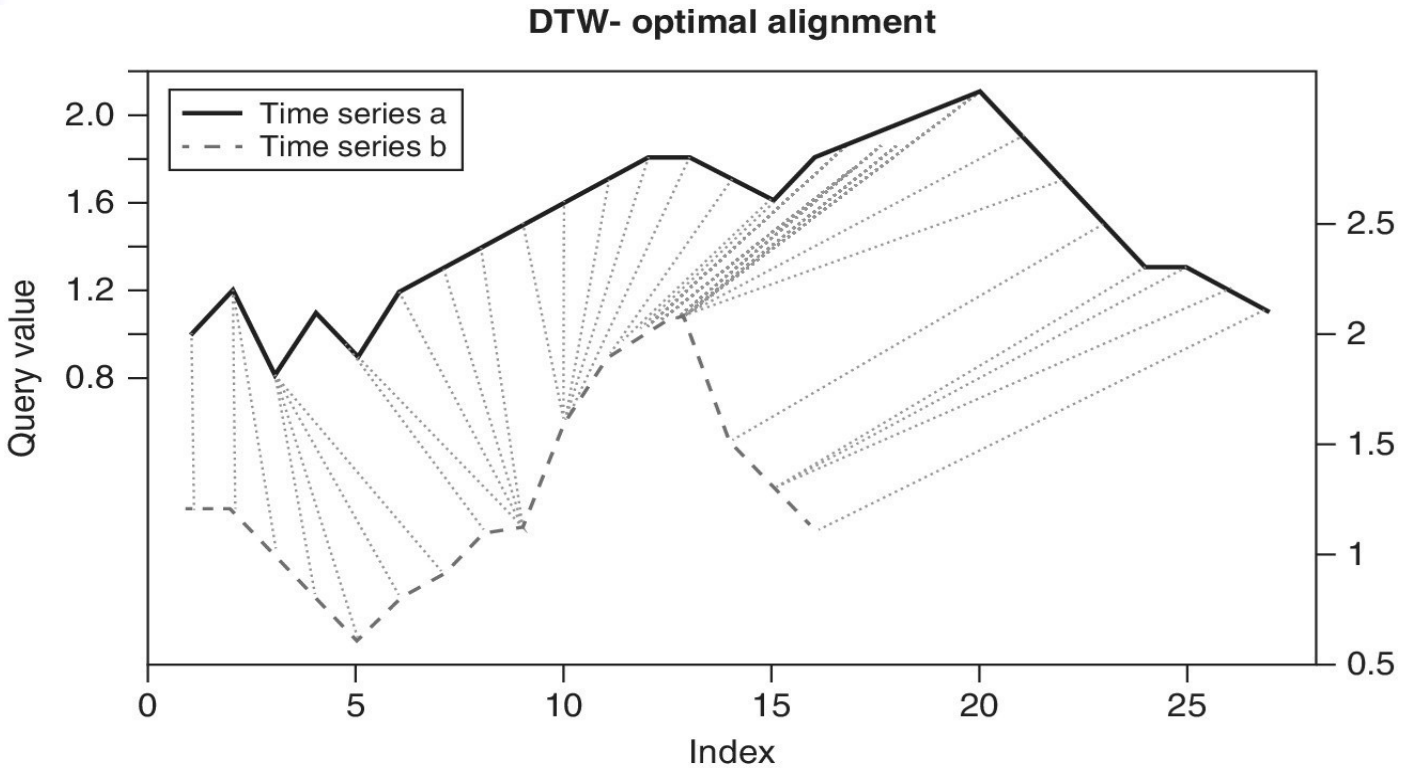
	I	will	go	to	the	party	but	first	have	work	they	by	bus
A	2	2	1	2	1	1	1	1	1	1	0	0	0
B	0	0	1	2	1	0	0	0	1	1	1	1	1

Difference between sequences

Dynamic Time Warping

- Calculates distance between time series data
- Returns the value of the best match between the two series, considering variations and shifts in time and speed
- The most optimal matching is calculated such that:
  - Each index from series a is matched with 1 or more index from series b
  - First and last index from series a is matched with the first and last index of series b
  - Monotonically increasing matches;
    if i, j are indices of *series a* such that $i < j$, and k, l are indices of *series b* such that $k < l$ then they **must not** be matched $i <-> l$ and $k <-> j$ (No crosses can occur in matching)
- Optimality is based on the pairwise similarity cost calculation
- Computationally heavy

Difference between sequences



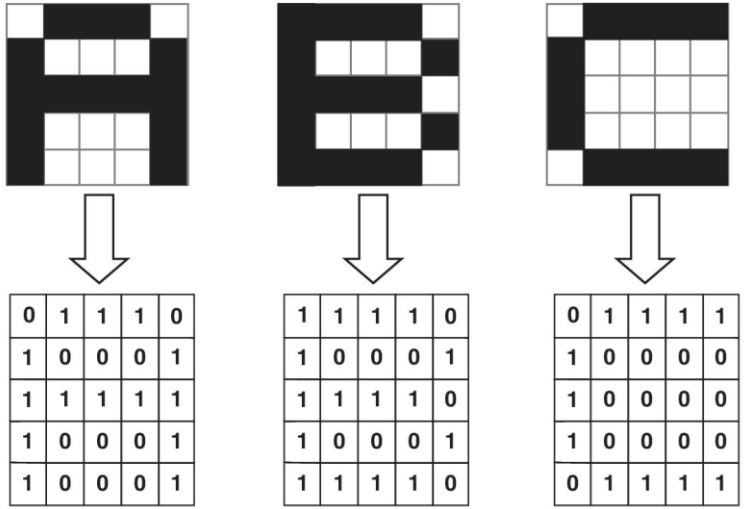
Distance between images

Features representation

- Features associated with the application are be extracted from the images
  - e.g. the distance between the eyes, number of hot pixels, image brightness etc.
- An image is represented by a vector of real numbers, where each element corresponds to one particular feature

Matrix/vector of pixels

- Each pixel or some area of pixels is converted into an integer value
- The size of the area is associated with the granularity required for the image



	1st row					2nd row					3rd row					4th row					5th row				
A	0	1	1	1	0	1	0	0	0	1	1	1	1	1	1	1	0	0	0	1	1	0	0	0	1
B	1	1	1	1	0	1	0	0	0	1	1	1	1	1	0	1	0	0	0	1	1	1	1	1	0
C	0	1	1	1	1	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1	1	1

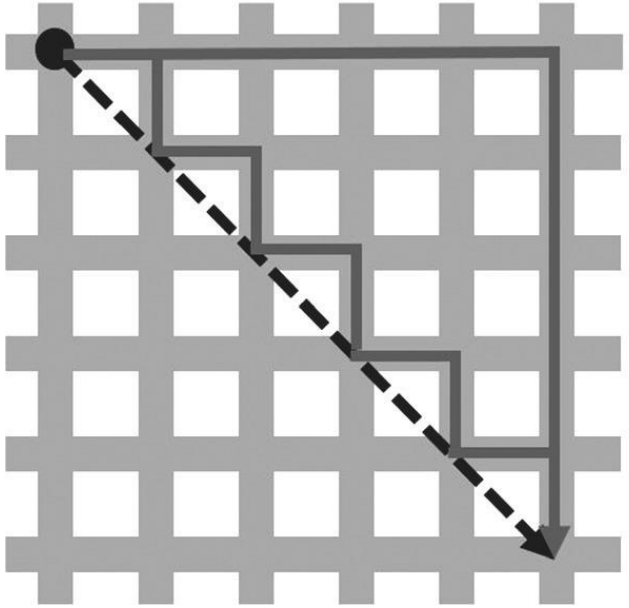
Multi-dimension distance measures

Minkowski distance

$$d(p,q) = \sqrt[r]{\sum_{k=1}^{m} |p_k - q_k|^r}$$

- Manhattan distance (r=1)
- Euclidean distance (r=2)

Usually, objects with mixed attribute types can be transformed to objects with only quantitative attributes.



Clustering techniques

Centroid Based

- Each object in the cluster is closer to a centroid representing the cluster than to a centroid representing any other cluster

Prototype-based

- Each object in the cluster is closer to a prototype representing the cluster than to a prototype representing any other cluster

Graph-based

- Represents the data set by a graph structure associating each node with an object and connecting objects that belong to the same cluster with an edge

Density-based

- A cluster is a region with high density of objects surrounded by a region of low density

K-means

- The most popular clustering algorithm
- Centroid/Partitional method
- Simple, good for convex clusters

DBSCAN

- Density-based method
- Excels in case of non-convex clusters

Agglomerative hierarchical clustering

- Easy-to-understand and interpret
- Graph-based method

K-means clustering

Input:
The dataset D,
the distance measure d,
the number K of clusters

Initialize the K centroids (cluster centers) randomly
repeat
    associate each instance in D with the
    closest centroid according to d
        recalculate each centroid using all
    instances from D associated with it
until no instance from D implies the change of its associated centroid

Centroid

- a prototype or profile of all the objects in a cluster, for example the average of all the objects in the cluster

Medoid

- the instance with the shortest sum of distances to the other instances of the cluster

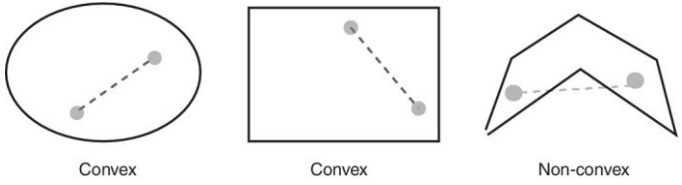
K-means clustering

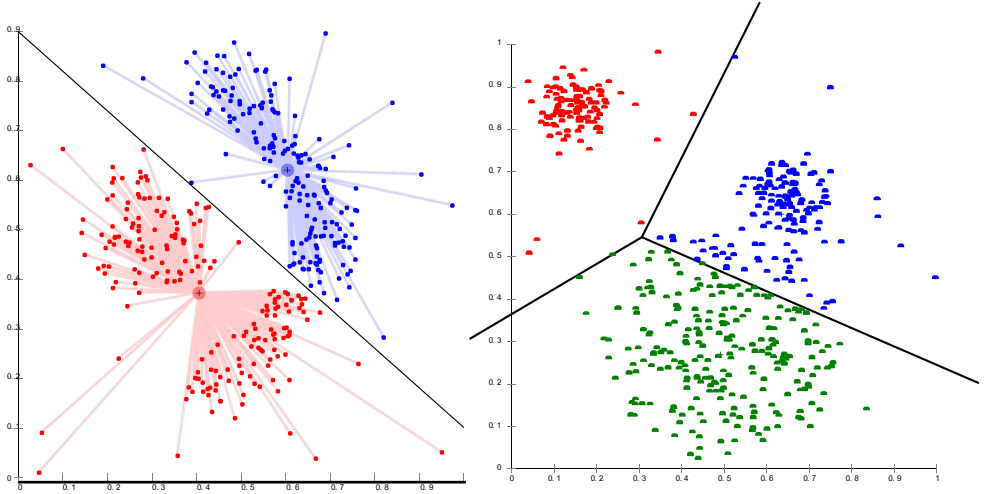
Pros

- Computationally efficient
- Often finds good results

Cons

- It might get stuck in local optima
  - Multiple runs with re-initializations
- Need to define K in the beginning
- Not good in case of noisy data and outliers
- Can find only convex shaped clusters





source: https://en.wikipedia.org/wiki/Cluster_analysis#Centroid-based_clustering

Optimal number of clusters

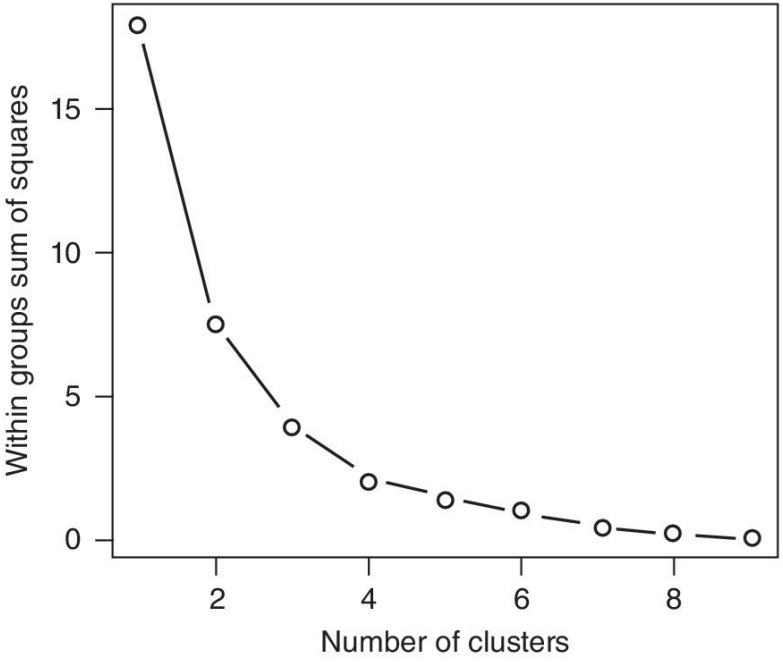
Within-groups sum of squares

- Sums up the squared Euclidean distance (sed) between each instance and the centroid of its cluster

$$s = \sum_{i=1}^{K} \sum_{j=1}^{J_i} sed(p_j, C_i)$$

  - where K is the number of clusters and J_i is the number of instances in cluster i, and C_i is the centroid of the cluster i.

- Elbow curve
  - Optimal K is at the elbow

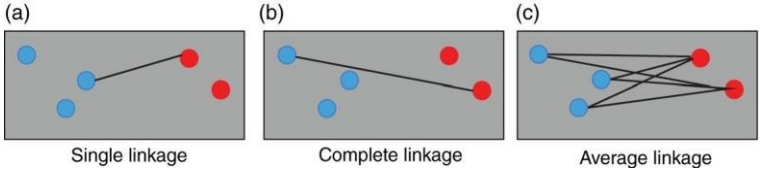


Agglomerative Hierarchical Clustering

Input:

The dataset D,

the linkage criterion



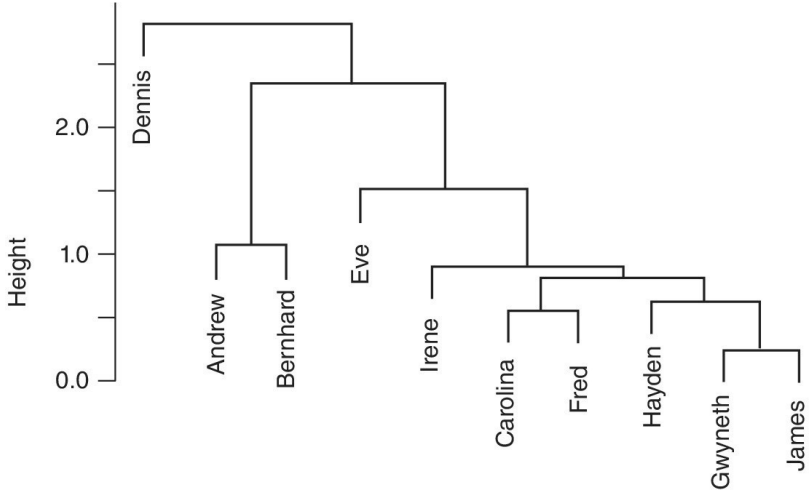
Initialize the clusters such that each object belongs to a separate cluster (Each object is its own cluster)

repeat

join the two most closest clusters according to the chosen linkage criterion

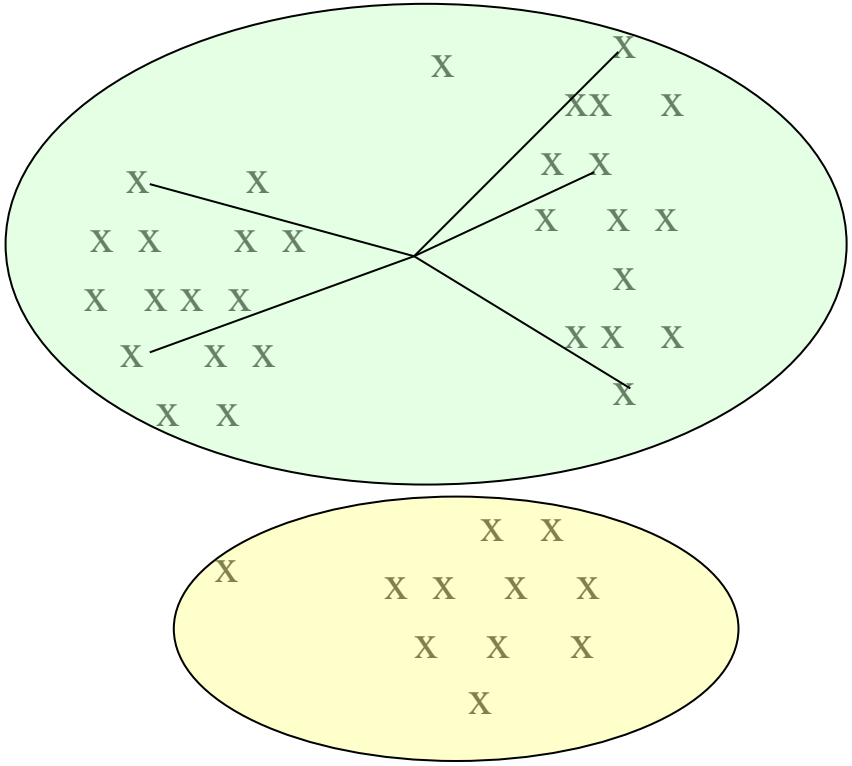
until the number of desired clusters is reached

Dendrograms



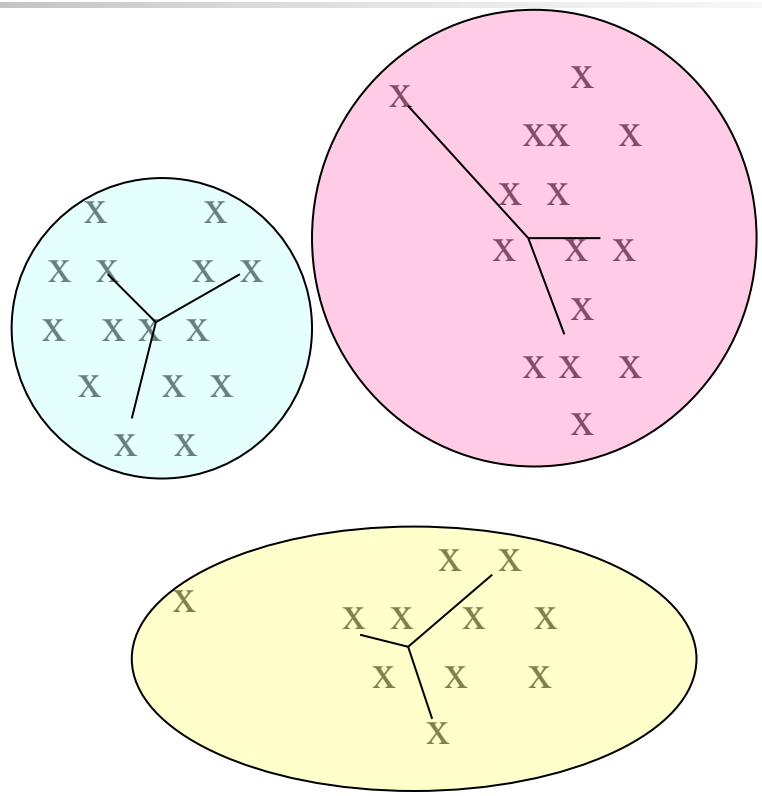
Example

Too few;
many long
distances
to centroid.



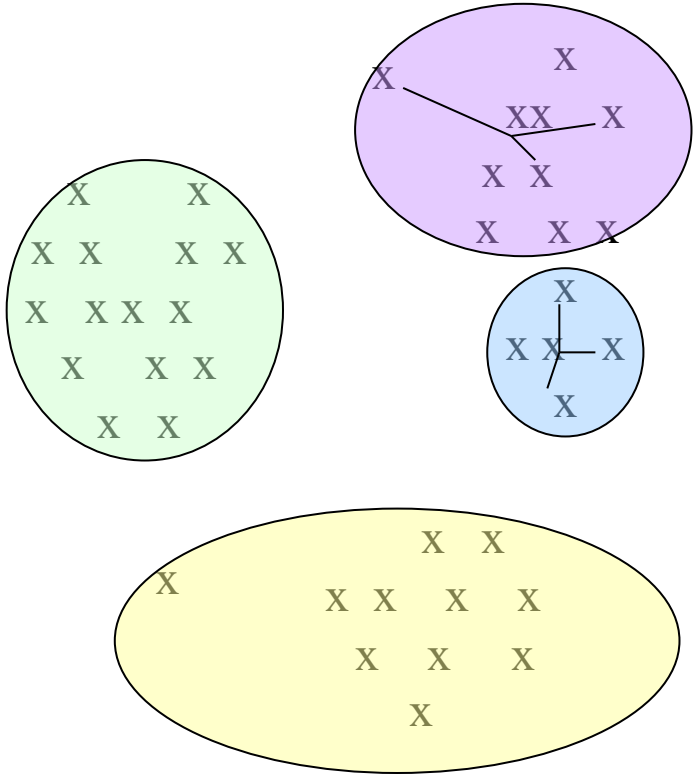
Example

Just right; distances rather short.

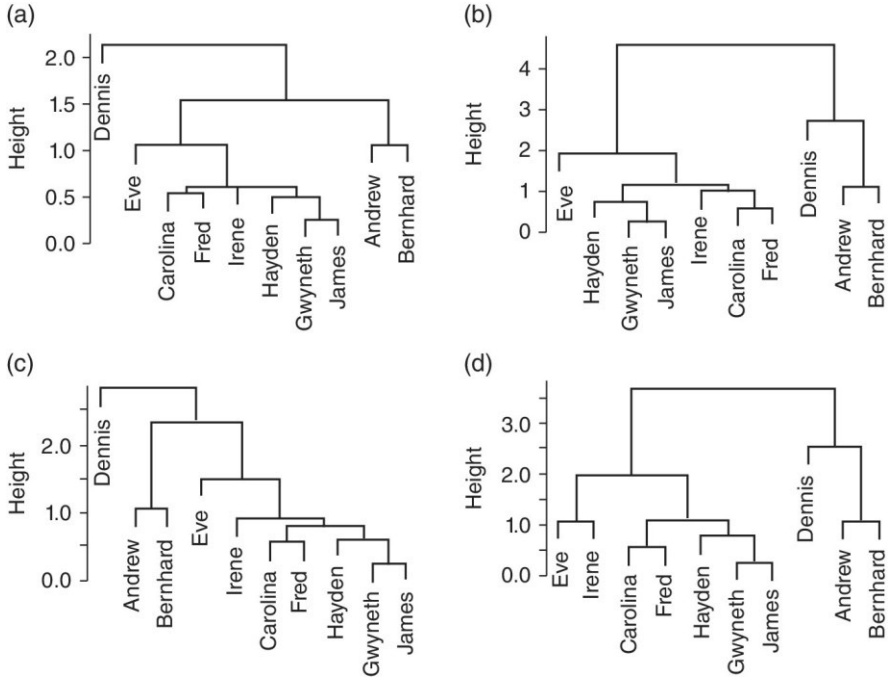


Example

Too many;
little improvement
in average
distance.



Agglomerative Hierarchical Clustering



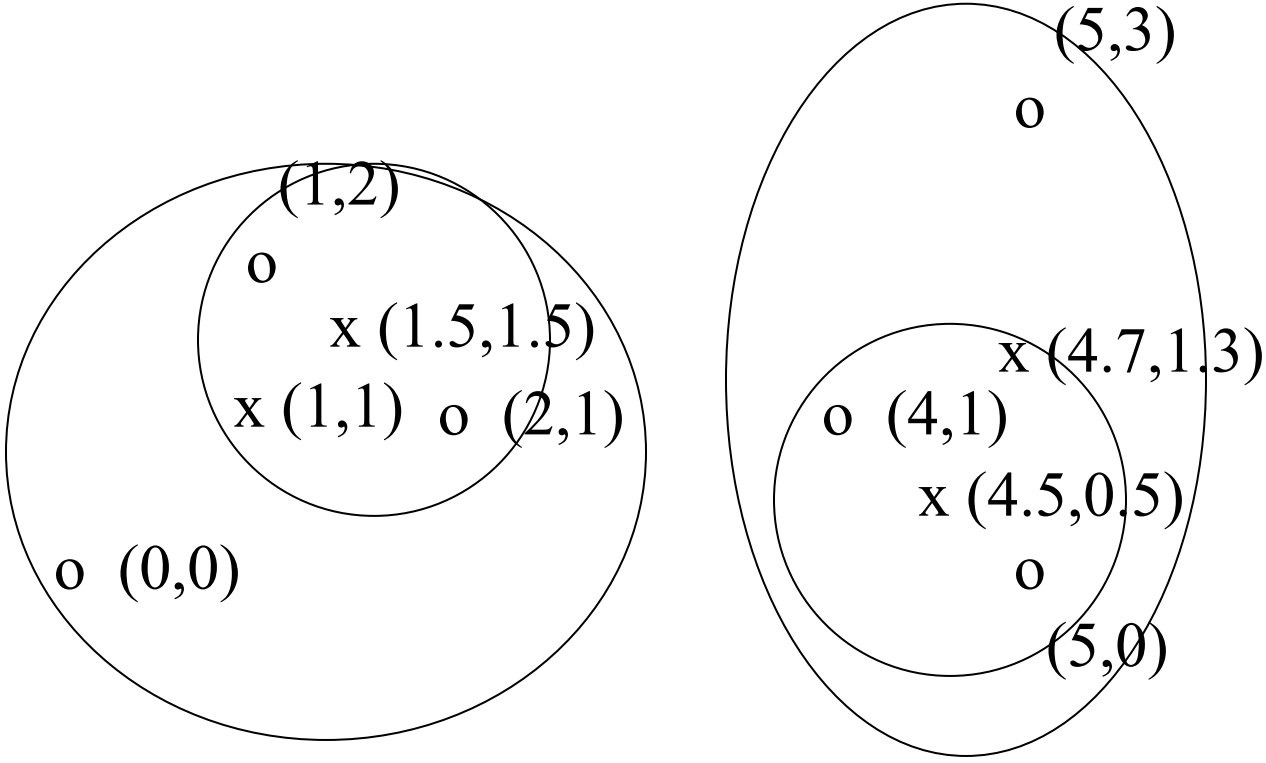
Pros

- Easily interpretable, but more confusing for large datasets
- Setting of hyper-parameter values is easy

Cons

- Computationally more complex than K-means
- Interpretation of dendrograms can be, in several domains, quite subjective
- Often gets stuck in local optima

Agglomerative Hierarchical Clustering



Evaluation of clusters

Silhouette

- Evaluates compactness inside clusters
  - How close to each other the objects inside a cluster are
  - How far the objects in each cluster are to the closest object from another cluster
- Meaning:
  - Silhouette value close to 1 means that x is in the centre of its cluster and far away from the neighboring clusters
  - Silhouette value close to -1 means that x is probably assigned to a wrong cluster
  - Silhouette value close to 0 means that x is on the boundary of two neighboring clusters
- The average over all objects is returned as the silhouette value for the whole clustering

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i,j)$$

$$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i,j)$$

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \text{ if } |C_i| > 1$$

Within-groups sum of squares

- Explained before

Final remarks

- Similarity measures have to be chosen carefully.
- Various techniques and even various launches of the same technique might end up with different results.
- Presence of correlated, irrelevant or redundant attributes can increase the computational cost and reduce the quality of the solutions found.
  - The lack of knowledge regarding class labels makes the identification of the relevant features in the feature selection process harder.

Frequent Pattern Mining

Transactional data

TID	Arabic	Indian	Mediterranean	Oriental	Fast Food
Andrew		✓	✓		
Bernhard		✓		✓	✓
Carolina		✓	✓	✓	
Dennis	✓		✓		
Eve				✓	
Fred		✓	✓	✓	
Gwyneth	✓		✓		
Hayden		✓		✓	✓
Irene		✓	✓	✓	
James	✓		✓		

Transactional data

TID	Items
Andrew	Indian, Mediterranean
Bernhard	Indian, Oriental, Fast food
Carolina	Indian, Mediterranean, Oriental
Dennis	Arabic, Mediterranean
Eve	Oriental
Fred	Indian, Mediterranean, Oriental
Gwyneth	Arabic, Mediterranean
Hayden	Indian, Oriental, Fast food
Irene	Indian, Mediterranean, Oriental
James	Arabic, Mediterranean

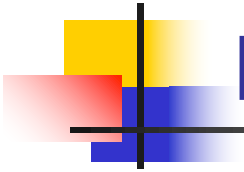
Itemsets and their support

- Itemset
  - An arbitrary combination of items
  - The number of itemsets grows exponentially with the number of items
- Support of an itemset
  - The frequency of an itemset in the data
  - Computed as the ratio of transactions containing all items from the itemset to the number of all transactions

Example

- *The support of itemset {Fast food} is 20% or 0.2*
- *Support({Indian, Oriental}) = 5/10 = 0.5 (50%)*

Number of items	Number of possible itemsets	Runtime in case of 1ms computation for 1 itemset
5	31	31 ms
10	1023	>1 sec
20	1048576	>17 min
30	1073741823	>12 days
40	$>10^{12}$	>34 years
50	$>10^{15}$	>35 millenia

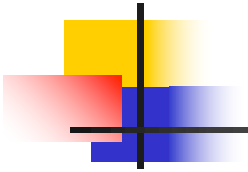


Frequent itemset mining

- Given
  - A set of all available items
  - Transactional data
  - And a minimum support threshold *min_sup*
- The goal is to find
  - Those (frequent) itemsets generated from items which support in the transactional data is at least *min_sup*
    - Low *min_sup* results in large number of itemsets which might be too specific
    - High *min_sup* results in a small number of itemsets which might be too generic

Example

- *1 frequent itemset for min_sup=0.7*
  - *{Mediterranean}*
- *4 frequent itemsets for min_sup=0.5*
  - *{Indian}, {Mediterranean}, {Oriental}*
  - *{Indian, Oriental}*
- *9 frequent itemsets for min_sup=0.3*
  - *{Arabic}, {Indian}, {Mediterranean}, {Oriental}*
  - *{Arabic, Mediterranean}, {Indian, Mediterranean}, {Indian, Oriental}, {Mediterranean, Oriental}*
  - *{Indian, Mediterranean, Oriental}*



Apriori

Input: Transactional data, *min_sup* threshold

set $k = 1$
set *stop* = false
repeat
    select all frequent itemsets of length k (with support at least *min_sup*)
    if there are no two frequent itemsets of length k **then**
        set *stop* = true
    else
        set $k = k + 1$
until *stop*

Further reading

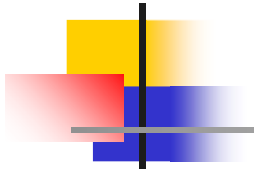
- **A General Introduction to Data Analytics**

  by João Mendes Moreira, André C. P. L. F. de Carvalho and Tomáš Horváth

- **Introduction to Data Mining**

  by Pang-Ning Tan, Michael Steinbach, Anuj Karpatne and Vipin Kumar

  https://www-users.cs.umn.edu/~kumar001/dmbook/index.php#item3



Thank You!