Clustering Algorithms

▼ *Types of Clustering Algorithms*

Types of Clustering Algorithms

Three types:

- Partitional Clustering Algorithms
  - Generates a single partition of the data to recover natural clusters
  - Input: Feature vectors
  - Examples: K-means, K-medoids
- Hierarchical Clustering Algorithms
  - Generates a sequence of nested partitions
  - Input: Distance matrix, which summarizes the distances between different feature vectors.
  - Example: Agglomerative clustering, divisive clustering
- Model-Based Clustering Algorithms
  - Assumes that the data is generated i.i.d from a mixture of distributions, each of which determines a different cluster.
  - Example: Expectation-Maximixation (EM)

▼ *Partitional Clustering Algorithm*

▼ *How to calculate inertia of clusters*

Partitional Clustering Algorithms: The Basics

- Goal: Assign N observations into K clusters, where $K < N$, that ensure high intra-cluster similarity and low inter-cluster similarity.
- Can be formulated as a combinatorial optimization problem.
- This requires defining a measure of intra-cluster similarity.
- Notation:
  - $\mathcal{C}$ denote a clustering structure with K clusters
  - $C \in \mathcal{C}$ denote a component cluster in the clustering structure $\mathcal{C}$
  - $e \in C$ denote an example in cluster C

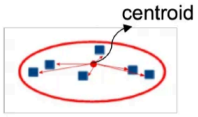
Measure of Intra-Cluster Similarity

Inertia of a Cluster

Inertia of a cluster C is defined as

$$\text{inertia}(C) = \sum_{e \in C} d_{\text{Euc}}(e, \text{centroid}(C))^2,$$

where $\text{centroid}(C)$ is the average of all examples in the cluster C.



- ***Inertia determines how compact the cluster is.***
  - ***The smaller the inertia***
    - ***The more similar the examples within the cluster***

Centroid calculated:

$\frac{x_1+x_2+x_3+x_4}{N}$, where N = 4, N representing the elements in a cluster $\therefore \Rightarrow$ $\frac{x_1+x_2+x_3+x_4}{4}$

Within Cluster Sum of Squares

Within Cluster Sum of Squares (WCSS) of a clustering structure $\mathcal{C}$ consisting of K clusters is defined as

$$\mathrm{WCSS}(\mathcal{C}) = \sum_{C \in \mathcal{C}} \mathrm{inertia}(C).$$

This quantifies the inertia of all clusters in the clustering structure.

If $WCSS(C) = \sum_{C \in C} inertia(C)$

and $inertia(C) = \sum_{e \in C} d_{Euc}(e, centroid(C))^2$

then logically $WCSS(C) = \sum_{c \in c} \sum_{e \in C} d_{Euc}(e, centroid(C))^2$ = $\sum_{c \in c} \sum_{e \in C} \sqrt{(e_1 - centroid(C)_1)^2 + \cdots + (e_m - centroid(C)_m)^2}$

▼ ***Optimisation Problem***

Optimization Problem

Find a clustering structure $\mathcal{C}$ of K clusters that minimizes the following objective:

$$\begin{aligned} & \min_{\mathcal{C}} \mathrm{WCSS}(\mathcal{C}) \\ \text{s.t} \quad & C_1 \cup C_2 \cup \ldots \cup C_K = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\} \quad (5) \\ & C_k \cap C_{k'} = \emptyset, \quad \forall k \neq k'. \quad (6) \end{aligned}$$

The condition (5) ensures that each example belongs to at least one of the K clusters. The condition (6) ensures that the cluster are non-overlapping, i.e., no example belongs to more than one cluster.

In this optimization problem,

- WCSS is an unnormalized quantity: Larger clusters with high inertia are penalized more than smaller clusters with high inertia.
- Minimizing WCSS of a clustering structure equivalently maximizes the inter-cluster dissimilarity. (will see this in detail later)

Problem with trying to solve the optimization problem:

- Finding the exact solution of the above problem is prohibitively hard i.e it's intractable
  - There are almost K^N ways to partition N observations into K clusters
  - Infeasible when N is large
- Hence
  - Look for sub-optimal approximate solutions via iterative algorithms
    - Like
      - K-means
      - K-medoids

Example 2: Revisiting

Assume that a partitional clustering algorithm on the mammal data returns two clusters: C_1 =(Dog, Cow) and C_2 =(Badger, Bear, Fox). Using the normalized data matrix, calculate the cluster centroids.
Solution:

- Centroid of C_1:

$$\Big((0+1)/2, (0+1)/2, (0+1)/2, (1+0.5)/2, (1+5/395)/2\Big)$$
$$= (0.5, 0.5, 0.5, 0.75, 200/395).$$

- Try for cluster C_2

▼ ***K-means***

Recap: Goal of Partitional Clustering Algorithms

Recall that the partitional clustering problem can be framed as the following optimization problem:

Clustering as an optimization problem

Find a clustering structure $\mathcal{C}$ of K clusters that minimizes the following objective

$$\begin{aligned} & \min_{\mathcal{C}} \mathrm{WCSS}(\mathcal{C}) \\ \text{s.t} \quad & C_1 \cup C_2 \cup \ldots \cup C_K = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\} \\ & C_k \cap C_{k'} = \emptyset, \quad \forall k \neq k', \end{aligned} \tag{1}$$

where

$$\mathrm{WCSS}(\mathcal{C}) = \sum_{C \in \mathcal{C}} \mathrm{intertia}(C) = \sum_{C \in \mathcal{C}} \sum_{e \in C} d_{\mathrm{Euc}}(e, \mathrm{Centroid}(C))^2.$$

K-Means

- K-means is an iterative, greedy descent algorithm that aims to find a sub-optimal solution to the optimization problem in (1).
- Specifically, K-means iteratively alternate between the following two steps:
  - **Assignment Step**: For a given set of K cluster centroids, K-means assigns each example to the cluster with closest centroid.

    $$\mathrm{WCSS}(\mathcal{C}) = \sum_{C \in \mathcal{C}} \sum_{e \in C} d_{\mathrm{Euc}}(e, \underbrace{\mathrm{Centroid}(C)}_{\text{fixed}})^2.$$

    This is equivalent to fixing the centroids and optimizing the cluster assignment to ensure low WCSS.
  - **Refitting step**: Re-evaluate and update the cluster centroids.

    This is equivalent to fixing the cluster assignment and optimizing the centroids to ensure low WCSS.
  - Consequently, over iterations, WCSS continue to decrease.

The Algorithm

K-Means Algorithm

Input: Number K of clusters and N examples $(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)})$.

1. Select K of the N examples at random as centroids and label it as $c_1, \ldots, c_K$.
2. Repeat until cluster centroids do not change:
   - (Assignment Step) Form K clusters by assigning each observation to its closest cluster centroid, i.e.,

$$\text{Cluster}(\mathbf{x}^{(i)}) = \arg \min_{k=1,\ldots,K} d_{\text{Euc}}(\mathbf{x}^{(i)}, c_k)^2, \quad \text{for } i = 1, \ldots, N$$

   - (Refitting Step) Compute the centroid of the obtained K clusters as

$$c_k = \frac{1}{N_k} \sum_{i:\text{Cluster}(\mathbf{x}^{(i)})=k} \mathbf{x}^{(i)}, \quad \text{for } k = 1, \ldots, K,$$

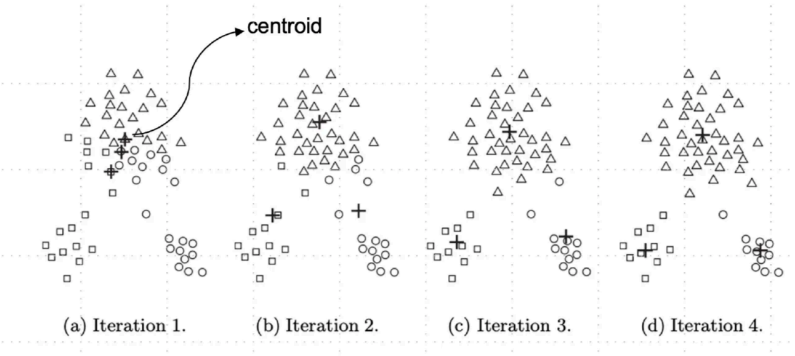
     where N_k is the total number of examples in the kth cluster.

The centroid is recomputed each time,except the beginning as the mean of the elements in the cluster

Important points:

- *Initial choices of cluster centroids are **randomly** chosen from the dataset*
- *However*
  - *Centroids found by K-means over the iterations need not be data examples*
    - *This is because cluster assignment is based on squared Euclidean distance*
- *The stopping criterion [singular of criteria]*
  - *When cluster centroids don't change*

Illustration



Note that cluster centroids need not be examples in later iterations.

In my opinion most likely iteration 5 will make the clusters perfec

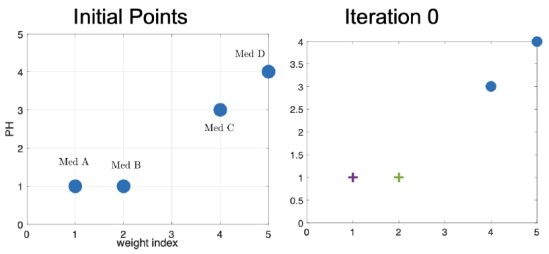
▼ ***Example***

Recap and Introduction

Example 1: Clustering of Medicines

Problem: Use K-means algorithm to cluster the following medicines into $K = 2$ clusters.

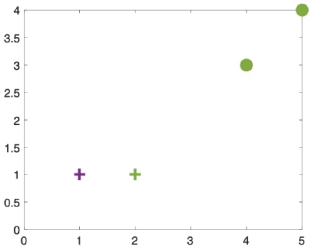
	Weight Index	PH
Med A	1	1
Med B	2	1
Med C	4	3
Med D	5	4



Iteration 0: Let initial centroids of cluster 1 and cluster 2 be chosen as Med A and Med B respectively, i.e., $c_1 = (1, 1)$ and $c_2 = (2, 1)$.

Iteration 1:

1. Calculate the squared Euclidean distance of each point to cluster centroids to form an object-centroid distance matrix:



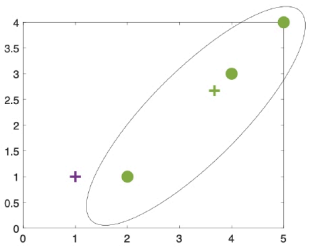
	Med A	Med B	Med C	Med D
c_1	0		13	25
c_2		0	8	18

$$d_{\text{Euc}}(MedC, c_1)^2 = (4-1)^2 + (3-1)^2 = 13$$
$$d_{\text{Euc}}(MedC, c_2)^2 = (4-2)^2 + (3-1)^2 = 8$$
$$d_{\text{Euc}}(MedD, c_1)^2 = (5-1)^2 + (4-1)^2 = 25$$
$$d_{\text{Euc}}(MedD, c_2)^2 = (5-2)^2 + (4-1)^2 = 18$$

Based on the matrix above, assign each medicine to its closest centroid. Thus, medicines B, C and D are assigned to Cluster 2.

2. Update the centroids of the clusters obtained in the previous step. This results in

$$c_1 \leftarrow c_1$$
$$c_2 \leftarrow \frac{1}{3}(\text{MedB} + \text{MedC} + \text{MedD}) = \left(\frac{2+4+5}{3}, \frac{1+3+4}{3}\right) = (3.67, 2.67).$$



Iteration 2:

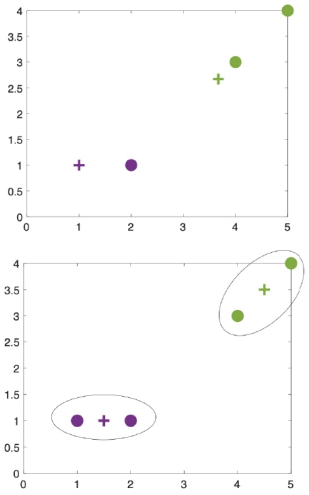
1. Calculate distance of each point to new cluster centroids.

	Med A	Med B	Med C	Med D
c_1	0	1	13	25
c_2	9.92	5.58	0.22	3.53

Med B is thus moved to cluster 1.

2. Update the centroids of the cluster.

$$c_1 \leftarrow \frac{1}{2}(\text{MedA} + \text{MedB}) = \left(\frac{1+2}{2}, \frac{1+1}{2}\right) = (1.5, 1)$$
$$c_2 \leftarrow \frac{1}{2}(\text{MedC} + \text{MedD}) = \left(\frac{4+5}{2}, \frac{3+4}{2}\right) = (4.5, 3.5).$$



Iteration 3:

- Repeat the same steps as before.
- Note that cluster assignments do not change.
- Algorithm has converged.

Space and Time Complexity of K-Means

- Space requirement for K-means is modest because only data observations and centroids are stored.
- Storage complexity is of the order $O((N + K)m)$ where m is the number of feature attributes.
- Time complexity of K-means is of the order $O(IKNm)$ where I is the number of iterations required for convergence.
- Importantly, time complexity of K-means is linear in N.

💡 ***In emphasis of the 3^{rd} bullet point:***

I is a constant

Number of iterations required for convergence

K is a constant

Number of centroids/clusters

m is a constant

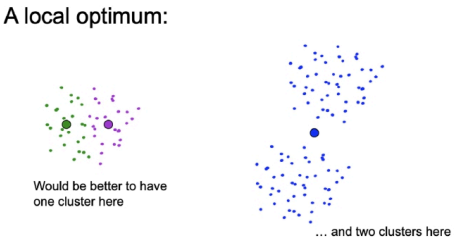
Number of feature attributes

💡 **Important questions**

- ***Does the K-means algorithms always converge?***
- ***Can it ALWAYS find optimal clustering?***
- ***How should we initialize the algorithm?***
- ***How should we choose the number of clusters?***

Convergence of K-Means

- At each iteration, the assignment and refitting steps ensure that the WCSS in (1) monotonically decreases.
- Moreover, K-means work with finite number (K^N) of partitions of the data.
- The above two conditions ensure that the K-means algorithm always converge.
- However, the optimization problem of (1) is non-convex. As such, K-means algorithm may not converge to the global minimum, but to a local minimum.

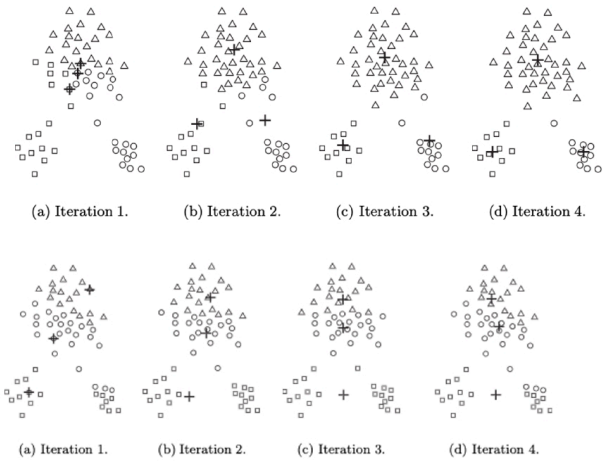


Escaping local minima: multiple random restarts and choose the clustering with lowest WCSS.

Therefore with K-means, findings the global minimum is not a guaranteed certainty

Choice of Initial Cluster Centroids

- Choosing initial cluster centroids is crucial for K-means algorithm.
- Different initializations may lead to convergence to different local optima.
- K-means is a **non-deterministic** algorithm.



Solutions

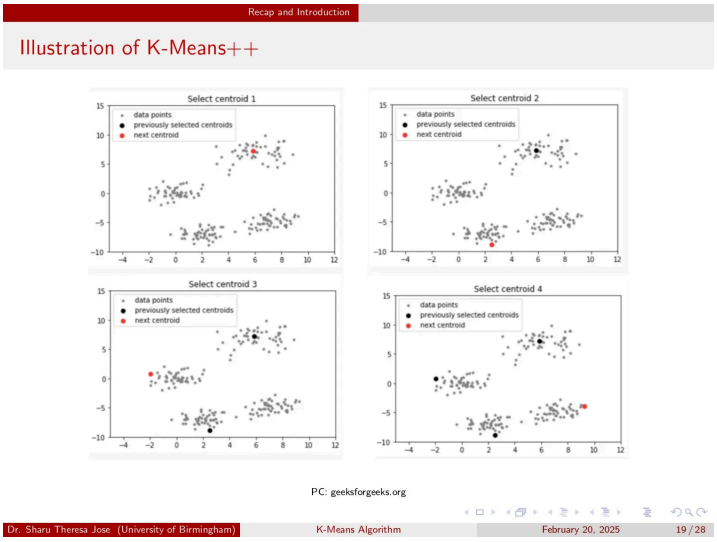
- Run multiple K-means algorithm starting from randomly chosen cluster centroids. Choose the cluster assignment that has the minimum WCSS.
- Specialized initialiation strategies: K-means ++

▼ ***K-means ++***

K-means ++ Algorithm

- Choose first centroid c_1 at random.
- Repeat until K centroids are found:
  - For each data point $\mathbf{x}$, compute the distance $d_{\text{Euc}}(\mathbf{x}, c) = d(\mathbf{x})$ from its nearest centroid c.
  - Choose a new data point $\mathbf{x}$ randomly with probability proportional to $d(\mathbf{x})^2$ as the next centroid. (Thus, a data point $\mathbf{x}$ that is farthest from the nearest centroid is highly likely to be picked as the next centroid)
- Use the obtained K centroids as initial centroids for the K-means algorithm.
- Run the K-means.

All this algorithm is adding on top of the K-means is trying select the initial centroids still "randomly" but trying to ensure each centroid picked is as far away as possible from each other maximising to chance of the convergence of data points into clusters to be the global minimum instead of a local minimum



Choice of the Number K of Clusters

- Conventional approach: Use prior domain knowledge
  - Example: Data segmentation - a company determines the number of clusters into which its employees must be segmented.
- A data-based approach for estimating the number of natural clusters in the dataset: **Elbow Method**

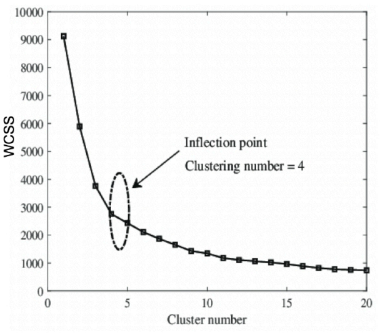
Elbow Method

- Run K-means algorithm repeatedly for increasing values of K.
- Evaluate the WCSS of the obtained clustering structure in each run of K-means.
- Plot WCSS($\mathcal{C}$) as a function of the number K of clusters.
- Optimal K lies at the elbow of the plot.

Some drawbacks of the elbow method:

- Heuristic concept
- Not easy to identify the elbow

Elbow Method



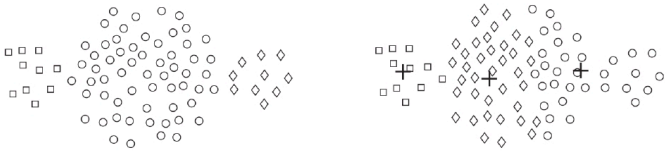
Intuition behind elbow method:

- Assume that there are actually K^* distinct clusters in the observed dataset.
- For $K < K^*$, clusters returned by K-means will each contain a subset of the true underlying clusters.
  - Consequently, WCSS tends to decrease **substantially** with each successive increase in K (until it reaches K^*) as natural groups are successively assigned to separate clusters.
- For $K > K^*$, one of the clusters estimated by K-means must partition at least one of the natural groups into two.
  - This will provide **smaller decrease** in WCSS as K is further increased.
- $K = K^*$ corresponds to the number of clusters where there is sharp decrease in successive differences in WCSS. This point thus corresponds to the elbow or kink in the plot.

The point of inflection on the graph is the elbow point

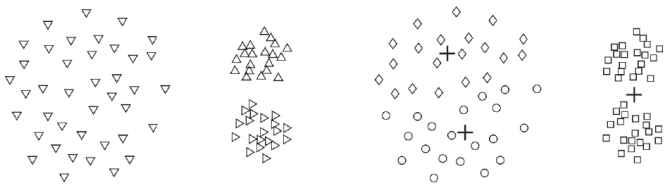
Other limitations of K-Means

- K-means has problems when outliers are present. Outliers can influence the clusters that are found and can also increase the WCSS.
- K-means has problems when clusters are of differing
  - Sizes



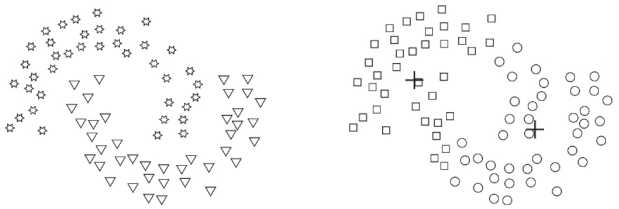
(a) Original points. (b) Three K-means clusters.

- K-means has problems when clusters are of differing
  - Densities



(a) Original points. (b) Three K-means clusters.

  - Non-globular shapes



(a) Original points. (b) Two K-means clusters.

One solution is to find a large number of clusters such that each of them represents a part of a natural cluster. But these small clusters need to be put together in a post-processing step.

▼ ***Application***

A practical application of K-Means

K-means algorithm is a key tool in the area of image and signal compression, particularly in **vector quantization**.

- (Left photo) 1024×1024 pixel image, where each pixel is a greyscale value ranging from 0 to 255. Requires 8 bits per pixel, and 1 megabyte of storage.
- (Center photo) Vector quantization (VQ)-based compressed version of left photo. Requires 0.239 of the storage of the left photo.
- (Right photo) Even more compressed version. Requires only 0.0625 of the storage of the left photo.

Vector Quantization via K-means

- Break left image into small blocks of 2×2. This results in 512×512 blocks of four numbers, each regarded as a (feature) vector in $\mathbb{R}^4$.
- Run K-means algorithm in the resulting space of feature vectors. The clustering process is called the **encoding step** in VQ, and the collection of centroids is called the **codebook**.
- Center image uses $K = 200$ while the right one uses $K = 4$.
- To represent the compressed image, approximate each of the 512×512 blocks by its closest **cluster centroid**, known as **codeword**. Then, construct the image from the centroids, a process called **decoding**.
- Storage requirements:
  - For each block, the identity of the cluster centroid need to be stored.
  - This requires $\log_2(K)$ bits per block, which equals $\log_2(K)/4$ bits per pixel.

Summary

Properties of K-means:

- Optimizes a global objective function
- Based on squared Euclidean distance
- Non-deterministic

Challenges of K-means:

- Requires as input the number K of clusters and an initial choice of centroids.
- Convergence to local minima implies multiple restarts.

▼ ***Hierarchical Clustering Algorithm***

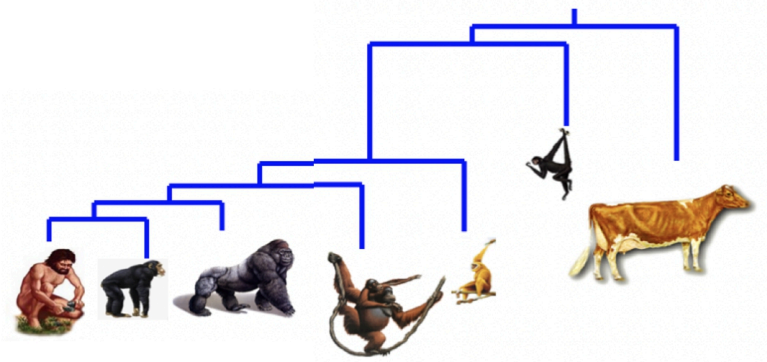
- Understand the difference between hierarchical and partitional clustering algorithms.
- Apply hierarchical clustering to problems and visualise the results.
- Interpret the obtained clustering structure

- Introduction to Hierarchical Clustering
- Agglomerative Hierarchical clustering
- Inter-Cluster Dissimilarity Metrics
- Characteristics of Hierarchical Clustering

- Recall: K-Means algorithm requires the user to supply the number K of required clusters and an initial choice of centroids.
- Hierarchical clustering requires no such specifications.
- Instead, user is only required to specify a measure of similarity (or dissimilarity) between a pair of clusters.

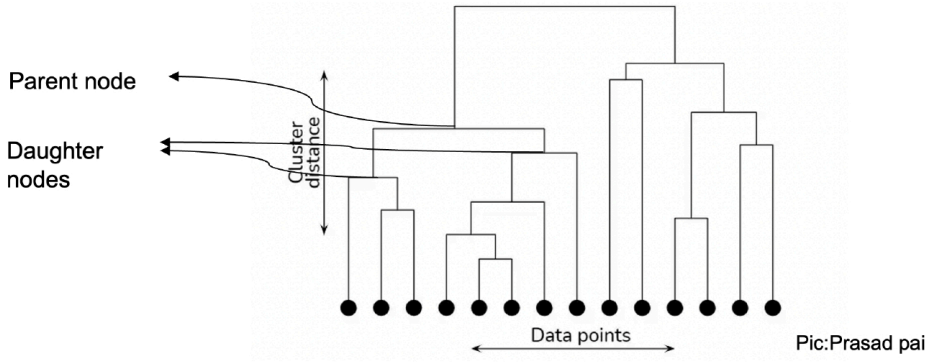
What is Hierarchical Clustering?

- Creates a hierarchical decomposition of the set of examples using a user-specified criterion
- Produces a **dendrogram**



Dendrogram

- Highly interpretable complete description of the hierarchical clustering in a graphical format
- Representation of hierarchical clustering as a rooted binary tree
- Nodes of the trees represent clusters



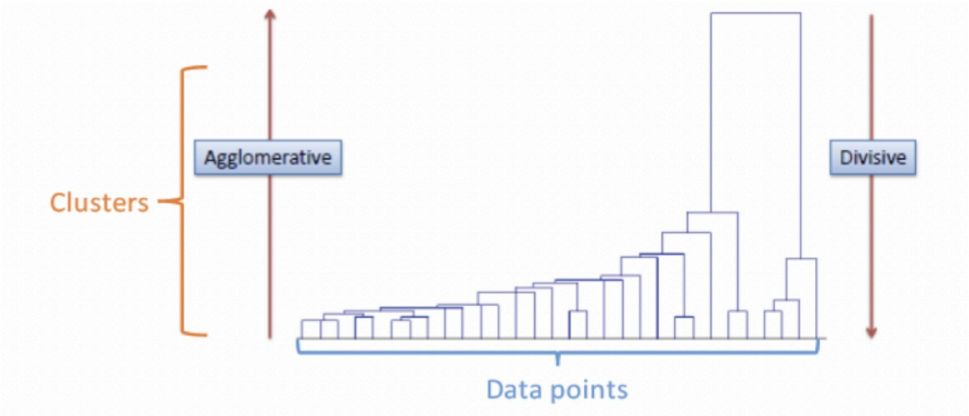
Strategies for Hierarchical Clustering

Agglomerative Clustering

- Bottom-up approach
- Starts at the bottom with each cluster containing a single observation
- At each level up, recursively merge pair of clusters with the smallest inter-cluster dissimilarity into a single cluster.
- A single cluster at the top level

Divisive Clustering

- Top-down approach
- Starts at the top with a single cluster of all observations
- At each level down, recursively split one of the existing clusters into two new clusters with the largest inter-cluster dissimilarity.
- At the bottom, each cluster contains single observation



Agglomerative Clustering Algorithm

Algorithm

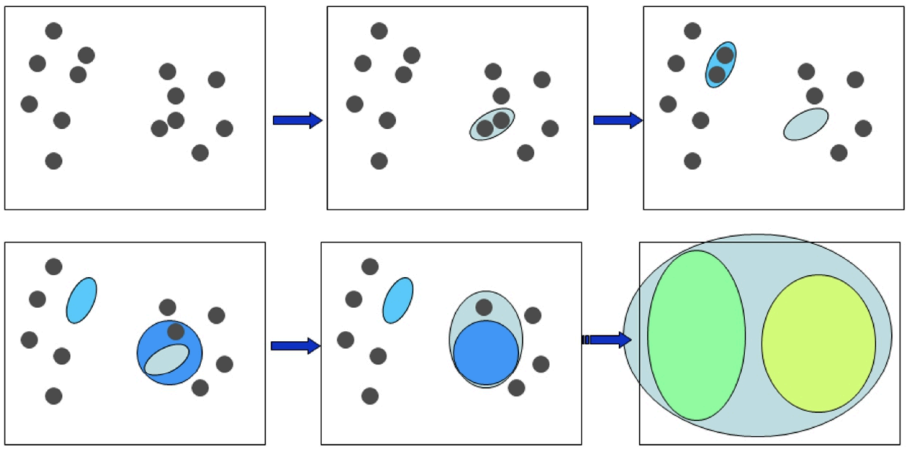
Input: For N examples, an $N \times N$ distance matrix summarizing the distance between each pair of examples.

Input: A user-specified measure of inter-cluster dissimilarity $d(C_i, C_j)$ between two clusters C_i and C_j.

1. Start with N clusters each consisting of one example.
2. Repeat until only one cluster remains:
   1. Find 2 clusters C_1 and C_2 that are most similar, or equivalently, have the smallest inter-cluster dissimilarity $d(C_1, C_2)$.
   2. Merge C_1 and C_2 into one cluster.

Output: A Dendrogram

An Illustration



Input 1: Distance Matrix

What is distance matrix?

- Given N observations $(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)})$ of examples/feature vectors, **distance matrix** summarizes the similarity relationship among the N observations.
- Distance matrix D is an $N \times N$ matrix (matrix with N rows and N columns) whose entry in ith row and jth column is given by

$$D_{i,j} = d(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}),$$

where $d(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$ is a given distance measure (e.g., Euclidean, Manhattan, Chebychev etc.).

- Properties of distance matrix:
  - Symmetric: $D_{i,j} = D_{j,i}$
  - Zero-diagonal entries: $D_{i,i} = 0$.

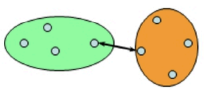
Input 2: Inter-Cluster Dissimilarity Metrics

Single Linkage (SL)

SL distance is the shortest distance from any member of the cluster to any member of the other cluster. For two clusters C_1 and C_2,

$$d_{\mathrm{SL}}(C_1, C_2) = \min_{i \in C_1, j \in C_2} d(i, j),$$

where $d(i, j)$ denotes a distance measure (eg., Euclidean, Manhattan etc.) between example i in cluster C_1 and example j in cluster C_2.



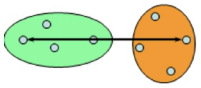
- Agglomerative clustering with single-linkage merges two clusters with the smallest single-linkage distance between them in each step.

Input 2: Inter-Cluster Dissimilarity Metrics

Complete Linkage (CL)

CL distance is the largest distance from any member of the cluster to any member of the other cluster:

$$d_{\mathrm{CL}}(C_1, C_2) = \max_{i \in C_1, j \in C_2} d(i, j).$$



- Agglomerative clustering with complete-linkage merges two clusters with the smallest complete-linkage distance between them in each step.

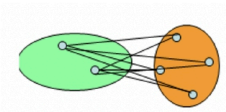
Input 2: Inter-Cluster Dissimilarity Metrics

Group Average (GA)

GA distance is the average distance between members of the two clusters:

$$d_{\mathrm{GA}}(C_1, C_2) = \frac{1}{N_{C_1} N_{C_2}} \sum_{i \in C_1, j \in C_2} d(i, j),$$

where N_{C_1} and N_{C_2} denote the number of examples in cluster C_1 and C_2 respectively.



- Agglomerative clustering with group average merges two clusters with the smallest group average distance between them in each step.

Comparison between the linkages

Single Linkage

- SL is determined by the pair of examples in the two clusters that are the closest; dissimilarities between other pairs of examples in the clusters do not matter.
- Consequently, SL induces **chaining effect**: tendency to combine clusters linked by a series of close intermediate examples.
- Results in clusters that are not compact

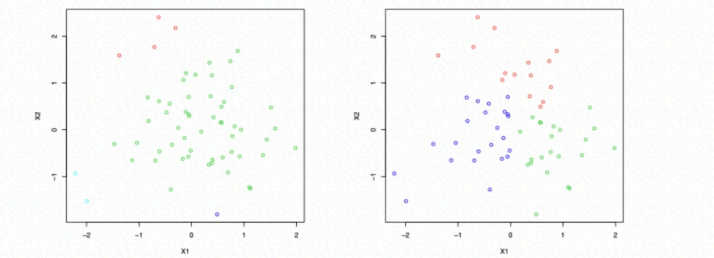
Complete Linkage

- Requires all examples in the two clusters to be relatively similar
- Produces compact clusters with small diameters
- However, an example in a CL-linkage based merged cluster can be closer to examples in other clusters than to examples in its own cluster. This induces **crowding** of clusters with clusters not far enough apart.

Group Average

- Attempts to produce relatively compact clusters that are relatively far apart
- Results of group average clustering can change with a monotone increasing transformation of the distance measures.

Illustration of Chaining and Crowding



Example 1: Clustering of European Cities Based on Air Distance

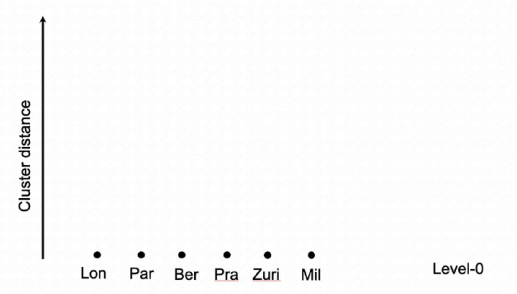
Given the distance matrix as below, obtain a **single-linkage dendrogram**.

	Lond	Paris	Berlin	Prague	Zurich	Milan
Lond	0	393	932	1027	776	958
Paris		0	878	883	489	641
Berlin			0	279	650	795
Prague				0	528	401
Zurich					0	204
Milan						0

Solution

Level 0:

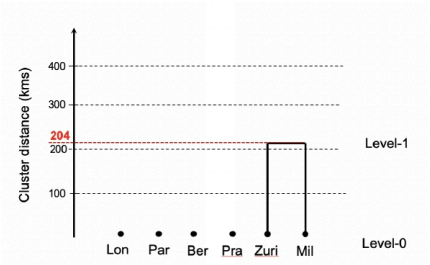
Clusters: (Lond), (Paris), (Berlin), (Prague), (Zurich), (Milan)



Level 1:

	Lond	Paris	Berlin	Prague	Zurich	Milan
Lond	0	393	932	1027	776	958
Paris		0	878	883	489	641
Berlin			0	279	650	795
Prague				0	528	401
Zurich					0	204
Milan						0

- Clusters (Milan) and (Zurich) have the smallest SL distance of 204. Merge them.
- New Clusters: (Lond), (Paris), (Berlin), (Prague), (Zurich, Milan)



In the dendrogram, height at which two clusters merge corresponds to their inter-cluster dissimilarity distance.

Level 2:

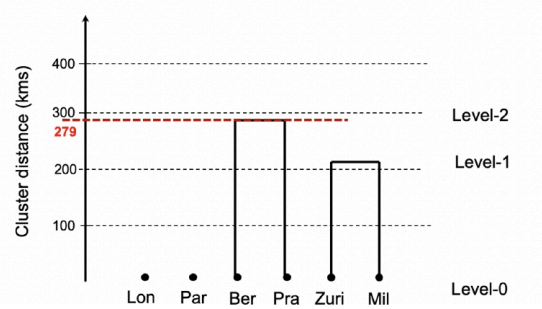
	Lond	Paris	Berlin	Prague	(Zurich, Milan)
Lond	0	393	932	1027	?
Paris		0	878	883	?
Berlin			0	279	?
Prague				0	?
(Zurich, Milan)					0

Compute the following SL distances:

$$d_{\mathrm{SL}}(Lond, (Zurich, Milan)) = \min\{d(Lond, Zurich), d(Lond, Milan)\} = \min\{776, 958\} = 776$$
$$d_{\mathrm{SL}}(Paris, (Zurich, Milan)) = \min\{d(Paris, Zurich), d(Paris, Milan)\} = \min\{489, 641\} = 489$$
$$d_{\mathrm{SL}}(Berlin, (Zurich, Milan)) = \min\{d(Berlin, Zurich), d(Berlin, Milan)\} = \min\{650, 795\} = 650$$
$$d_{\mathrm{SL}}(Prague, (Zurich, Milan)) = \min\{d(Prague, Zurich), d(Prague, Milan)\} = \min\{528, 401\} = 401$$

	Lond	Paris	Berlin	Prague	(Zurich, Milan)
Lond	0	393	932	1027	776
Paris		0	878	883	489
Berlin			0	279	650
Prague				0	401
(Zurich, Milan)					0

- Clusters (Berlin) and (Prague) have the smallest SL linkage distance (=279). Merge them.
- New clusters: (Lond), (Paris), (Berlin, Prague), (Zurich, Milan)



Level 3:

	Lond	Paris	(Berlin, Prague)	(Zurich, Milan)
Lond	0	393	?	776
Paris		0	?	489
(Berlin, Prague)			0	?
(Zurich, Milan)				0

The SL distances can be computed as:

$$d_{\mathrm{SL}}(Lond, (Berlin, Prague)) = \min\{932, 1027\} = 932$$
$$d_{\mathrm{SL}}(Paris, (Berlin, Prague)) = \min\{878, 883\} = 878$$
$$d_{\mathrm{SL}}((Zurich, Milan), (Berlin, Prague)) = \min\{650, 528, 795, 401\} = 401.$$

The last calculation

$d_{SL}((Zurich, Milan), (Berlin, Prague)) = min\{650, 528, 795, 401\} = 401$

uses the original distance matrix

Whereas the other 2 calculations:

$$d_{SL}(Lond, (Berlin, Prague)) = min\{932, 1027\} = 932$$

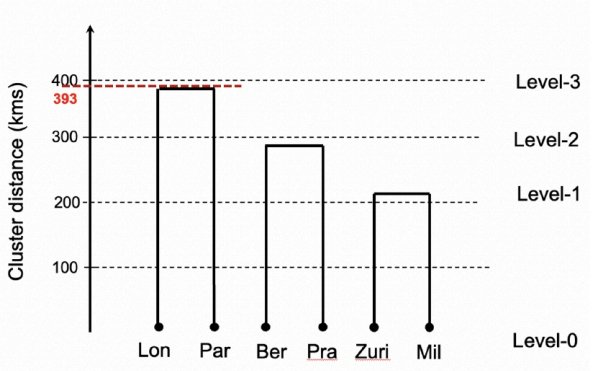
$$d_{SL}(Paris, (Berlin, Prague)) = min\{878, 883\} = 878$$

use the updated distance matrix.

This results in:

	Lond	Paris	(Berlin, Prague)	(Zurich, Milan)
Lond	0	393	932	776
Paris		0	878	489
(Berlin, Prague)			0	401
(Zurich, Milan)				0

- Clusters (Lond) and (Paris) have the smallest SL linkage. Merge them.
- New clusters: (Lond,Paris), (Berlin, Prague), (Zurich, Milan)

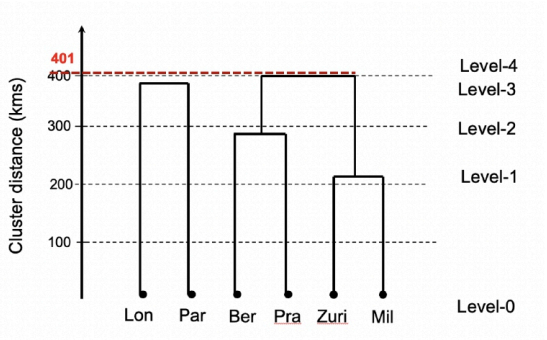


Level 4:

For the new clusters, again compute the SL distances. This results in the following matrix.

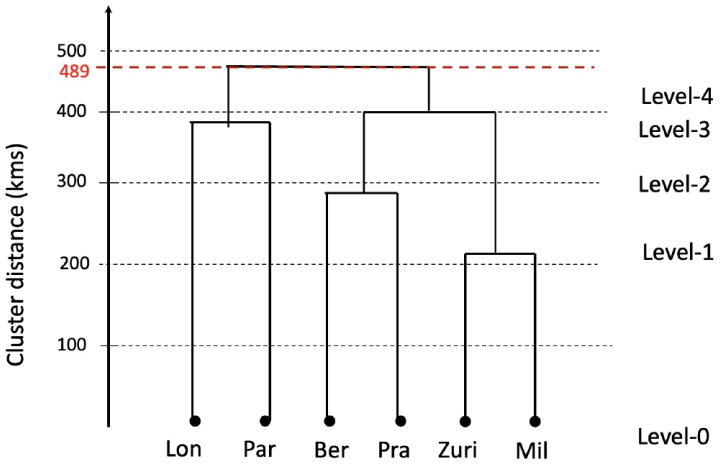
	(Lond, Paris)	(Berlin, Prague)	(Zurich, Milan)
(Lond,Paris)	0	878	489
(Berlin, Prague)		0	401
(Zurich, Milan)			0

- Clusters (Berlin, Prague) and (Zurich, Milan) have the smallest SL distance (=401). Merge them.
- New clusters: (Lond, Paris), (Berlin, Prague, Zurich, Milan)



Level 5:

- Finally, the two clusters (Lond, Paris) and (Berlin, Prague, Zurich, Milan) merge.
- The height at which they merge is given by the $d_{\mathrm{SL}}((Lond, Paris), (Berlin, Prague, Zurich, Milan)) = 489$.



Homework Question

For the same example, try implementing complete linkage and group average agglomerative clustering algorithms. Compare the obtained dendrograms.

Reading a Dendrogram

- Height at which two clusters merge corresponds to their inter-cluster dissimilarity distance
- Possesses a monotonicity property, i.e., inter-cluster dissimilarity between merged clusters is monotonically increasing with the level of the merger.
- Horizontally cutting dendrogram at a particular height partitions observations into disjoint clusters.

Space and Time Complexity

- Storage Complexity: $O(N^2)$
  - Storing the distance matrix requires storage of $N^2/2$ entries.
- Time Complexity is $O(N^3)$ in many cases.
  - There are N iterations, and in each iteration the N^2-size distance matrix needs to be updated and searched.
  - Complexity can be reduced to $O(N^2 \log(N))$ time for some approaches.
- Space and time complexity severely limits the size of datasets that can be processed.

Characteristics of Hierarchical Clustering

- Lack of a global objective function
  - Need not solve hard combinatorial optimization problem as in K-means
  - No issues with local minima or choosing initial points
- Deterministic algorithm
- Merging decisions are final. But may impose a hierarchical structure on an otherwise un-hierarchical data.

▼ ***Evaluation of Clustering Algorithms***

Learning Outcomes

- Understand the importance of cluster analysis
- Familiarize with some commonly used cluster validation criteria

Overview of Lecture

- Introduction to Cluster Evaluation
- Cluster Validation Criteria
  - Unsupervised, Supervised and Relative Validation Criteria
- Unsupervised and Supervised Validation Criteria

Introduction

- Supervised learning has well-accepted evaluation measures and procedures (e.g., accuracy, cross-validation).
- In contrast, cluster evaluation (or validation) is not trivial.
- Nevertheless, cluster validation is important: every clustering algorithm will find clusters in a dataset, even if data has no natural clustering structure.

Cluster Validation Can Help Answer...

- Is there a clustering tendency in the observed data, i.e., determine whether non-random structure (or natural grouping) exists in the data?
- Can we evaluate how well the results of a clustering algorithm fit the data (or natural grouping) **without** external information?
- Can we evaluate how well the results of a clustering algorithm fit the data **with** external information?
- Can we compare two sets of clusters to determine which is better?
- Can we determine the correct number of clusters?

What is Cluster Validation?

- Goal: Evaluate in a quantitative and objective manner the cluster structure found by an algorithm according to a validation criterion
- Validation criterion: Index used to measure the adequacy of the found cluster structures.
- Adequacy refers to the sense in which the found cluster structure provides true information about the data or reflect the intrinsic character of the data.

Types of Cluster Validation Criteria

- **Unsupervised (Internal Indices)**
  - Measures goodness of a clustering structure **without** reference to external information
  - Example: WCSS
  - Can be further divided into two classes: intra-cluster and inter-cluster similarity indices.
  - Can also be used to estimate the optimal number of clusters (e.g., elbow method).
- **Supervised (External Indices)**
  - Measures the extent to which a clustering algorithm matches some externally supplied information.
  - Example: Entropy (how well cluster labels match with externally supplied class labels), also recall classes-to-cluster evaluation in Weka
- **Relative**
  - Compares two different sets of clusters or algorithms.
  - Can be a supervised or unsupervised criteria used for the purpose of comparison.
  - Example: Two K-means clusterings can be compared using either WCSS or entropy.

Unsupervised Validation Criteria

The following are examples of unsupervised validation criteria for partitional and hierarchical clustering algorithms.

- Partitional Clustering Algorithms
  - Variability and Separation-Based
  - Silhouette Coefficient
- Hierarchical Clustering Algorithms
  - Cophenetic Correlation

Variability and Separation Criteria

- Quantifies the inter-cluster (separation) and intra-cluster (variability) dissimilarity.
- Separation/variability criteria can be then used to define an overall validation criterion for a clustering structure $\mathcal{C}$.
- We have previously seen one measure of intra-cluster dissimilarity: Inertia.
- Recall: Inertia is the sum of the (squared) Euclidean distance of each example in the cluster to its centroid.

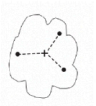
$$Inertia = \sum_{c \in C} \sum_{e \in C} d_{euc}(e, centroid(c))^2,$$

Where e is an element in the current cluster c and c is a cluster in the cluster structure C

- More generally, we can define the following measures of **centroid-based variability and separation** criteria:
  - Centroid-based variability of cluster C:

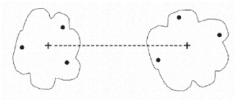
$$variability_c(C) = \sum_{e \in C} d(e, centroid(C))$$

where $d(\cdot, \cdot)$ is **any** distance measure. In particular, if $d(\cdot, \cdot)$ is the squared Euclidean distance, then this measure coincides with inertia.



  - Centroid-based inter-cluster separation between clusters C_1 and C_2:

$$separation_c(C_1, C_2) = d(centroid(C_1), centroid(C_2))$$



  - Centroid-based separation of cluster C_1 with respect to the whole data:

$$separation_c(C_1) = d(centroid(C_1), centroid(data)).$$

- The $variablity_c(C)$ is summing the distance measures between every point in the cluster with the centroid of that cluster
- The $seperation_c(C_1, C_2)$ is the distance measure between the centroids of the two clusters
- The $seperation_c(C_1)$ is the distance measure between the centroid of the focus cluster relative to the WHOLE dataset, including all the clusters being observed.

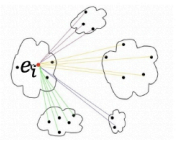
Silhouette Coefficient (SC)

- SC can be evaluated for an individual example, for a cluster, as well as for a clustering structure of K clusters.
- SC combines the ideas of variability and separation.
- **Computing SC for an example**: Let example e_i belongs to cluster C.
  - Calculate a_i = average distance of ith example to all other examples in its cluster, i.e.,

$$a_i = \frac{\sum_{e \in C, e \neq e_i} d(e_i, e)}{|C| - 1}.$$

  - Calculate b_i = minimum (over clusters) of the average distances of ith example to examples in another cluster, i.e.,

$$b_i = \min_{k=1,\ldots,K, C_k \neq C} \frac{\sum_{e \in C_k} d(e_i, e)}{|C_k|}$$



  - SC for ith example is

$$s_i = \frac{b_i - a_i}{\max\{b_i, a_i\}}$$

Okay this requires a little explanation:

So the first formula we have is

$$a_i = \frac{\sum_{e \in C, e \neq e_i} d(e_i, e)}{|C| - 1}$$

So this sums the distance from our chosen example to every other example in the cluster and then divides it by the total elements - 1, because our chosen example is exempt

The next formula is:

$$b_i = \min_{k=1,\ldots,K, C_k \neq C} \frac{\sum_{e \in C_k} d(e_i, e)}{|C_k|}$$

So this one sums the distance from our chosen example to every example in every cluster except the one the chosen example is in

The last formula is:

$$s_i = \frac{b_i - a_i}{\max(b_i, a_i)}$$

This one finds the range of the two sums and divides it by the maximum of the sums

Properties of SC

- SC can vary between -1 and 1.
  - $SC = -1$: $(a_i > b_i = 0) \implies$ data is better fit to a neighboring cluster
  - $SC = 0$: $(a_i = b_i) \implies$ data is on the border between two clusters
  - $SC = 1$: $(0 = a_i < b_i) \implies$ data is well-matched to the cluster
- SC of a cluster = average of SCs of examples in the cluster
- SC of a clustering = average of SC of all examples in the dataset.

🔥 1 is best

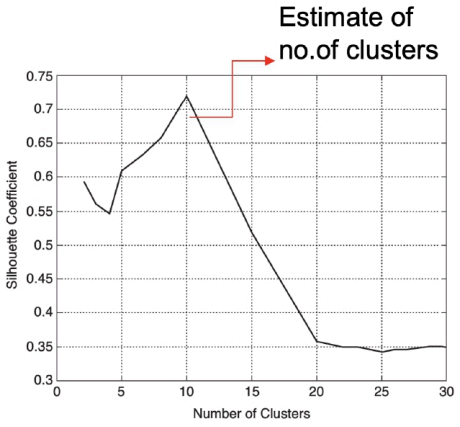
-1 is worst

0 is teetering on both [bordering two clusters]

SC to Estimate the Number of Clusters

Average SC of a clustering structure can be used to estimate the optimal number of clusters in the data set.

- Plot the average SC of clustering as a function of number of clusters
- Peak in the plot gives an estimate of the number of clusters.



Supervised Validity Criteria for Partitional Clustering Algorithms

- Supervised validation criteria make use of access to to external information in the form of externally derived class labels for data objects.
- For partitional clustering algorithms, there are two classes of supervised validation criteria:
  - **Classification-oriented**
    - Uses measures from classification
    - Quantifies the extent to which a cluster contains objects of a single class
    - Examples include: Entropy, Purity, Precision, Recall, F-measure
  - **Similarity-oriented**
    - Related to similarity measures for binary data
    - Quantifies the extent to which two objects in the same class are in the same cluster and vice versa
    - Examples include: Jaccard measure, Rand statistic
- We will focus on classification-oriented validation criteria.

Classification-Oriented Validity Measures

- Uses externally derived class labels for data examples.
- Example: Confusion matrix output of classes to clusters evaluation in WEKA on LA Times dataset. (Each entry corresponds to number of objects in a cluster that belongs to the corresponding class)

Predicted Cluster labels

True Class labels (externally supplied)	Cluster 1	Cluster 2	Cluster 3	Total
Entertainment	10	11	50	71
Finance	15	60	13	88
Foreign	20	21	9	50
Metro	3	15	2	20
National	45	2	11	58
Sports	12	28	56	96
Total	105	137	141	383

Confusion Matrix for the output of Clustering Algorithm on LA Times Dataset

- Let L denote the number of classes and K denote the number of clusters.
- Probability that an example of cluster i belongs to class j is given by

$$p_{i,j} = \frac{\text{number of examples of class j in cluster i}}{|C_i|}$$

Precision, Recall and F-Measure

- Precision of cluster i with respect to class j:

$$precision(i,j) = p_{i,j}$$

  - Measures the extent to which a cluster contains objects of a single class.
- Recall of cluster i with respect to class j:

$$recall(i,j) = \frac{\text{number of objects of class j in cluster i}}{\text{number of objects in class j}}$$

  - Determines the fraction of class j contained in cluster i
- F-measure of cluster i with respect to class j:

$$F(i,j) = \frac{2 * precision(i,j) * recall(i,j)}{precision(i,j) + recall(i,j)}$$

  - Measures the extent to which a cluster contains only objects of a particular class and all objects of that class.
  - Combination of both precision and recall.

$$precision(i,j) = p_{i,j} = \frac{n_{i,j}}{\sum_k n_{i,k}}$$

- $n_{i,j}$
  - The number of elements that belong to both cluster i and class j.
- $\sum_k n_{i,k}$
  - The total number of elements in cluster i, summing over all classes j in cluster i.

$$p_{i,j} = \frac{\text{The number of elements that belong to both cluster i and class j.}}{\text{The total number of elements in cluster i, summing over all classes j in cluster i.}}$$

Entropy

- Degree to which each cluster consists of examples of a single class
- Entropy of ith cluster:

$$ent(C_i) = -\sum_{j=1}^{L} p_{i,j} \log_2(p_{i,j})$$

- Total entropy of a set of clusters:

$$ent = \sum_{i=1}^{K} \frac{|C_i|}{\text{total number of examples}} ent(C_i)$$

- Low entropy $\implies$ clusters consists mostly of examples of same class.

$$ent(C_i) = -\sum_{j=1}^{L} \frac{n_{i,j}}{\sum_k n_{i,k}} \log_2(\frac{n_{i,j}}{\sum_k n_{i,k}})$$

$$ent(C_i) = -\sum_{j=1}^{L} \frac{\text{The number of elements that belong to both cluster i and class j.}}{\text{The total number of elements in cluster i, summing over all classes j in cluster i.}} \log_2(\frac{\text{The number of elements that belong to both cluster i and class j.}}{\text{The total number of elements in cluster i, summing over all classes j in cluster i.}})$$

Low entropy, examples of same class

High entropy, examples of several classes

Purity

- Another measure of the extent to which a cluster consists of examples of a single class.
- Purity of ith cluster:

$$Purity(C_i) = \max_j p_{i,j}$$

- Overall purity of the clustering structure:

$$Purity = \sum_{i=1}^{K} \frac{|C_i|}{\text{total no.of examples}} Purity(C_i)$$

- Ideally, we require high purity (close to 1).

So purity is the:

Across all clusters and for each cluster: $\left(\frac{\text{The number of elements in the cluster}}{\text{Total number of elements across all clusters}} * \text{maximum of}\left(\frac{\text{The number of elements that belong to both cluster i and class j.}}{\text{The total number of elements in cluster i, summing over all classes j in cluster i.}}\right)\right)$

Example

Consider the output of K-means clustering as summarized in the following table. Compute the entropy and purity of Cluster 1.

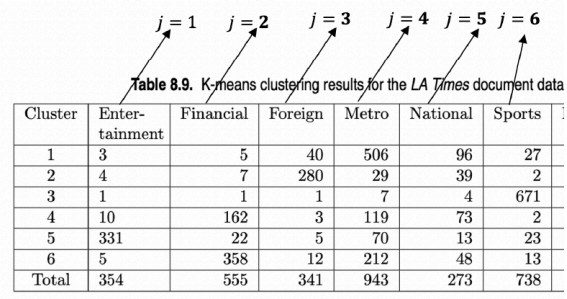


Table 8.9. K-means clustering results for the *LA Times* document data

Cluster	Entertainment	Financial	Foreign	Metro	National	Sports
1	3	5	40	506	96	27
2	4	7	280	29	39	2
3	1	1	1	7	4	671
4	10	162	3	119	73	2
5	331	22	5	70	13	23
6	5	358	12	212	48	13
Total	354	555	341	943	273	738

$$p_{1,1} = \frac{3}{3+5+40+506+96+27} = \frac{3}{677} = 0.0044$$

$$p_{1,2} = \frac{5}{677} = 0.0073, p_{1,3} = \frac{40}{677} = 0.0590$$

$$p_{1,4} = \frac{506}{677} = 0.7474, p_{1,5} = \frac{96}{677} = 0.1418,$$

$$p_{1,6} = \frac{27}{677} = 0.0398$$

Purity of cluster 1
$= \max_j p_{1,j} = 0.7474$
Entropy of Cluster 1
$= -\sum_j p_{1,j} \log_2(p_{1,j}) = 1.2270$

🔥 The purity here was calculated by taking the most dominant class as "class j". Purity always works like this