COMP809 Data Mining and Machine Learning

Patricio Maturana-Russel

p.maturana.russel@aut.ac.nz

Department of Mathematical Sciences and Computer Science and Software Engineering Departments, Auckland University of Technology, Auckland, New Zealand

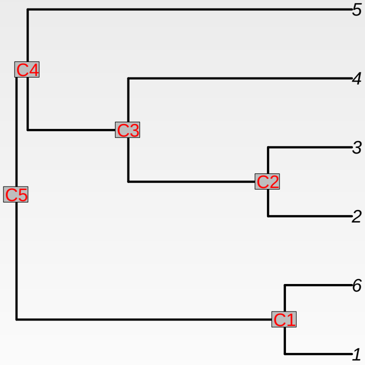
Semester 1, 2024

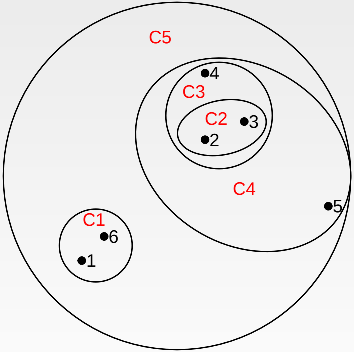
Contents

- Hierarchical Clustering
- Agglomerative
- Case study

Hierarchical Clustering

- It produces a set of nested clusters organized as a hierarchical tree.
- It can be visualized as a dendrogram–a tree like diagram that records the sequences of merges or splits.





Hierarchical Clustering

Two main types:

- Agglomerative (bottom-up):
  - Start with the **points as individual clusters**.
  - At each step, merge the closest pair of clusters until only one cluster (or K clusters) left.
- Divisive (top-down):
  - Start with one, **all-inclusive cluster**.
  - At each step, split a cluster until each cluster contains a point (or there are K clusters)

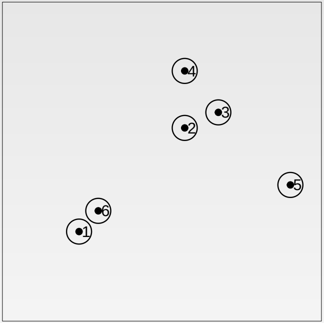
Traditional hierarchical algorithms use a similarity or distance matrix, and merge or split one cluster at a time.

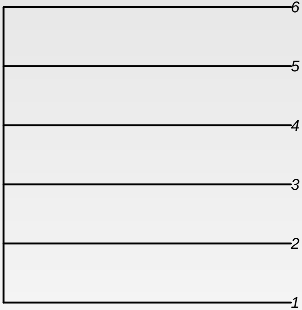
Agglomerative

- Most popular hierarchical clustering technique (simpler mathematically).
- Basic algorithm is straightforward:
  - Compute the proximity matrix
  - Let each data point be a cluster
  - **Repeat**
  - Merge the two closest clusters
  - Update the proximity matrix
  - **Until** only a single cluster remains
- Key operation is the computation of the proximity of two clusters.
  - Different approaches to defining the distance between clusters distinguish the different algorithms.

Agglomerative

Each data point is a cluster.



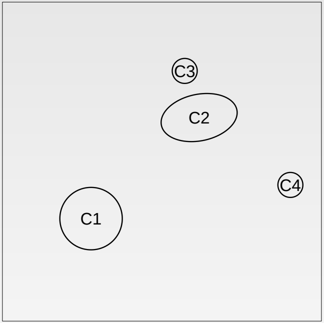


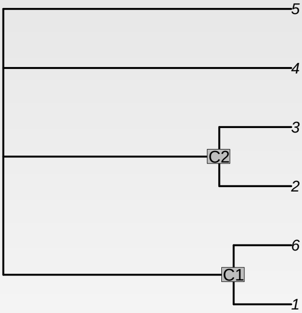
Proximity Matrix

	1	2	3	4	5	6
1						
2						
3						
4						
5						
6						

Agglomerative

After a few iterations:



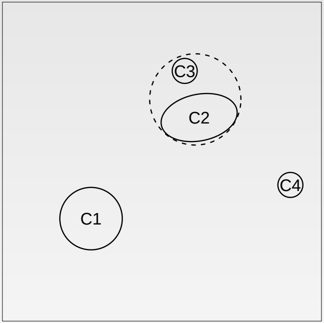


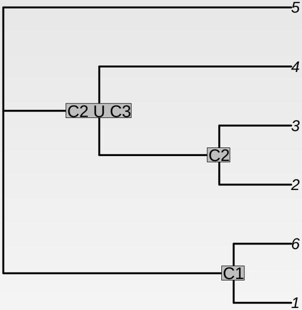
Proximity Matrix

	C1	C2	C3	C4
C1				
C2				
C3				
C4				

Agglomerative

Identify and merge the closest cluster

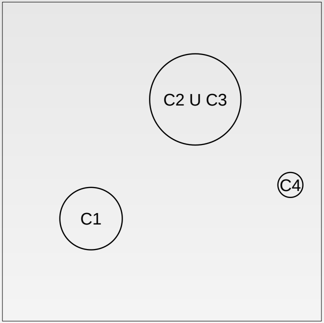


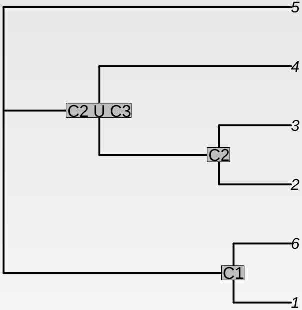


Proximity Matrix

	C1	C2	C3	C4
C1				
C2				
C3				
C4				

Agglomerative



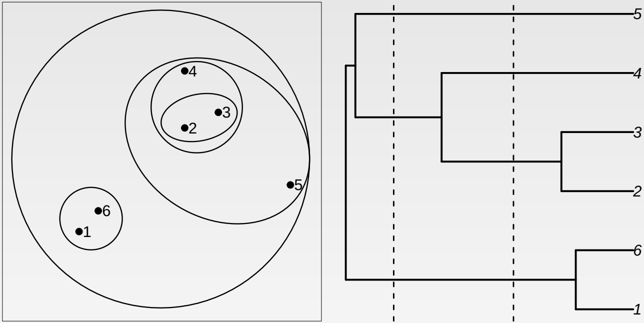


Proximity Matrix

	C1	C2 U C3	C4
C1			
C2 U C3			
C4			

Continue until one cluster remain.

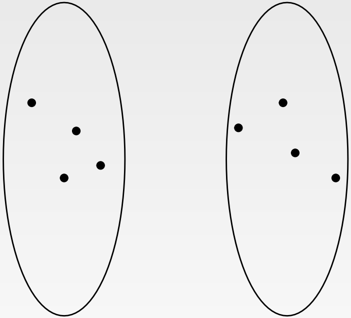
Agglomerative



Clustering obtained by cutting the dendrogram at a desired level: each connected component forms a cluster.

Inter-cluster similarity

- MIN (single-link)
- MAX (complete-link)
- Group Average (average-link)
- Distance Between Centroids (centroid)
- Ward

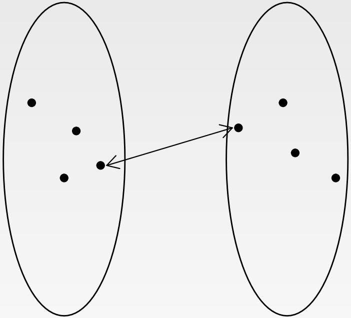


Inter-cluster similarity

- **MIN (single-link)**
- MAX (complete-link)
- Group Average (average-link)
- Distance Between Centroids (centroid)
- Ward

Limitations:

- Sensitive to noise and outliers.
- It produces long, elongated clusters.



Inter-cluster similarity

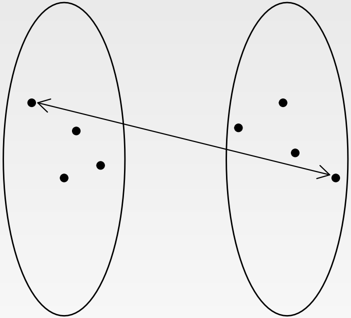
- MIN (single-link)
- **MAX (complete-link)**
- Group Average (average-link)
- Distance Between Centroids (centroid)
- Ward

Strength:

- More balanced clusters (with equal diameter).
- Less susceptible to noise.

Limitations:

- Tends to break large clusters.
- All clusters tend to have the same diameter–small clusters are merged with larger ones.



Inter-cluster similarity

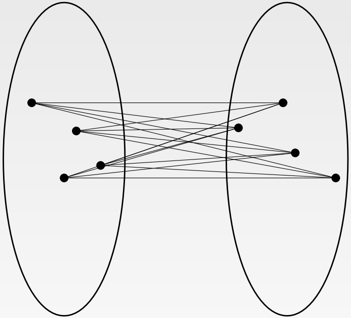
- MIN (single-link)
- MAX (complete-link)
- **Group Average (average-link)**
- Distance Between Centroids (centroid)
- Ward

Strength:

- Less susceptible to noise and outliers

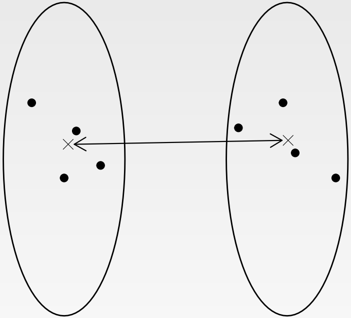
Limitations:

- Biased towards globular clusters



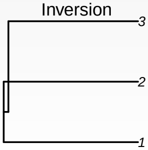
Inter-cluster similarity

- MIN (single-link)
- MAX (complete-link)
- Group Average (average-link)
- **Distance Between Centroids (centroid)**
- Ward



Limitations:

- It is not monotonic: smaller clusters can potentially be more similar to the new larger cluster than to their individual clusters causing an inversion in the dendrogram. It contradicts the fundamental assumption that small clusters are more coherent than large clusters. It does not arise in the other methods.



Inter-cluster similarity

- MIN (single-link)
- MAX (complete-link)
- Group Average (average-link)
- Distance Between Centroids (centroid)
- **Ward**

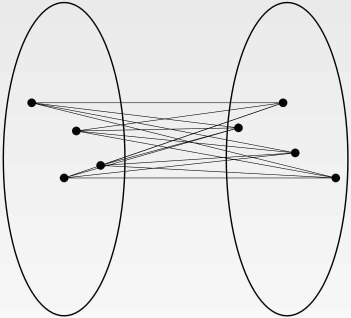
It minimizes the variance of the clusters being merged.

Strength:

- Less susceptible to noise and outliers.

Limitations:

- Biased towards globular clusters.



Inter-cluster similarity

Ward's distance between clusters C_i and C_j is defined as:

$$D_w(C_i, C_j) = \sum_{x \in C_i} (x - r_i)^2 + \sum_{x \in C_j} (x - r_j)^2 - \sum_{x \in C_{ij}} (x - r_{ij})^2,$$

where r_i, r_j, and r_{ij} are the centroids of C_i, C_j, and $C_{ij} = C_i \cup C_j$, respectively.

It is the difference between the sum of the squared errors of clusters C_i and C_j, and the squared error of the merging of both clusters.

Ward's method is the hierarchical analogue of K-means. It can be used to initialize K-means.

Hierarchical Clustering

Advantages:

- It shows all the possible linkages between clusters, which helps to understand the data.
- No need of assuming the number of clusters.
    - Any desired number of clusters can be obtained by 'cutting' the dendogram at the proper level.
- They may correspond to meaningful taxonomies.
    - Example in biological sciences (e.g., animal kingdom, phylogeny reconstruction, etc).

Hierarchical Clustering

Disadvantages:

- In large data sets
  - It can be difficult to interpret.
  - It can be computationally expensive.
- Once a decision is made to combine two clusters, it cannot be undone.
- No objective function is directly minimized.
- Different schemes have problems with one or more of the following:
  - Sensitivity to noise and outliers.
  - Difficulty handling different sized clusters and convex shapes.
  - Breaking large clusters.

In Python

First, we produce the dendogram to determine the number of clusters

```
>>> from scipy.cluster.hierarchy import dendrogram, linkage;
>>> dendrogram(linkage(data, method="ward"), orientation = "top",
              labels = None);
```

Find more information on www.docs.scipy.org

Then can get the clusters as follows:

```
>>> from sklearn.cluster import AgglomerativeClustering
>>> model = AgglomerativeClustering(n_clusters=2, linkage="ward")
>>> model.fit(data)
```

Find more information on `www.scikit-learn.org`

Case study

The dataset has been compiled from the United Nations Demographic Yearbook 1990 (United Nations publications) and has the following variables: *birth rate*, *death rate*, *infant death rate*, and *country*.

Can these variables be used to categorize these countries?

```
>>> import pandas as pd
>>> data = pd.read_csv("poverty.csv");
>>> print(data);
    Birth  Death  InfantDeath        Country
0    24.7    5.7         30.8        Albania
1    13.4   11.7         11.3 Czechoslovakia
2    11.6   13.4         14.8        Hungary
3    13.6   10.7         26.9        Romania
4    17.7   10.0         23.0           USSR
..    ...    ...          ...            ...
92   50.1   20.2        132.0        Somalia
93   44.6   15.8        108.0          Sudan
94   31.1    7.3         52.0        Tunisia
95   50.5   14.0        106.0       Tanzania
96   51.1   13.7         80.0         Zambia
```

Case study

```
>>> data[["Birth", "Death", "InfantDeath"]].describe();
              Birth        Death  InfantDeath
    count  97.000000   97.000000    97.000000
    mean   29.229897   10.836082    54.901031
    std    13.546695    4.647495    45.992584
    min     9.700000    2.200000     4.500000
    25%    14.500000    7.800000    13.100000
    50%    29.000000    9.500000    43.000000
    75%    42.200000   12.500000    83.000000
    max    52.200000   25.000000   181.600000
```

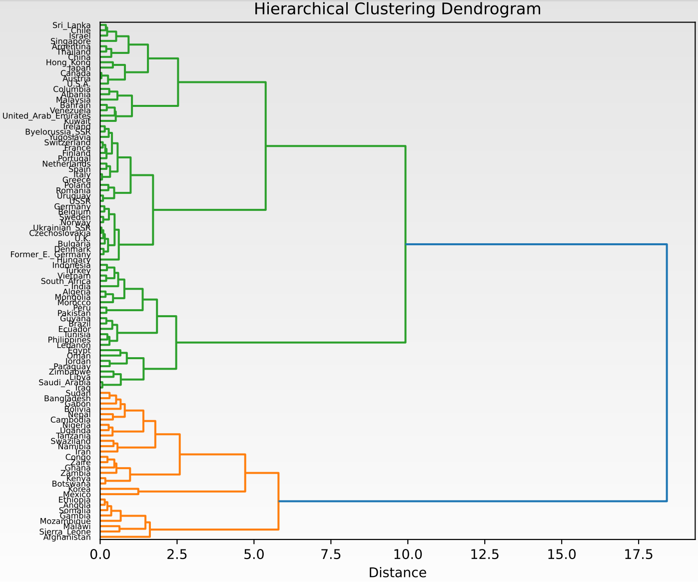
The SDs are quite different. The data will be standardized.

```
>>> from sklearn.preprocessing import StandardScaler
>>> X      = data.iloc[:,[0,1,2]];
>>> scaler = StandardScaler();
>>> fitted = scaler.fit(X)
>>> X_std  = pd.DataFrame(fitted.transform(X));
```

We can generate the dendrogram as follows:

```
>>> from scipy.cluster.hierarchy import dendrogram, linkage;
>>> dendrogram(linkage(X_std, method="ward"), orientation = "right",
               labels = data.Country.tolist());
```

Case study


Hierarchical Clustering Dendrogram
Sri_Lanka
Chile
Israel
Singapore
Argentina
Thailand
China
Hong_Kong
Japan
Canada
Austria
U.S.A.
Columbia
Albania
Malaysia
Bahrain
Venezuela
United_Arab_Emirates
Kuwait
Ireland
Byelorussia_SSR
Yugoslavia
Switzerland
France
Finland
Portugal
Netherlands
Spain
Italy
Greece
Poland
Romania
Uruguay
USSR
Germany
Belgium
Sweden
Norway
Ukrainian_SSR
Czechoslovakia
U.K.
Bulgaria
Denmark
Former_E._Germany
Hungary
Indonesia
Turkey
Vietnam
South_Africa
India
Algeria
Mongolia
Morocco
Peru
Pakistan
Guyana
Brazil
Ecuador
Tunisia
Philippines
Lebanon
Egypt
Oman
Jordan
Paraguay
Zimbabwe
Libya
Saudi_Arabia
Iraq
Sudan
Bangladesh
Gabon
Bolivia
Nepal
Cambodia
Nigeria
Uganda
Tanzania
Swaziland
Namibia
Iran
Congo
Zaire
Ghana
Zambia
Kenya
Botswana
Korea
Mexico
Ethiopia
Angola
Somalia
Gambia
Mozambique
Malawi
Sierra_Leone
Afghanistan
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Distance

Case study

The number of clusters can be inferred from the dendrogram by drawing a vertical line on it. This should be where we find the biggest distances. In our example, it could be between approximately

- 10 and 18, generating 2 clusters; or
- 5.5 and 10, generating 3 clusters.

Case study

We can add the cluster label to the original dataset for further analyses as follows:

```
>>> from sklearn.cluster import AgglomerativeClustering
>>> model = AgglomerativeClustering(n_clusters=3, linkage="ward",
                                    compute_distances = True);
>>> model.fit(X_std);

>>> data["Cluster"] = pd.DataFrame(model.labels_);
>>> print(data);
    Birth  Death  InfantDeath        Country  Cluster
0    24.7    5.7         30.8        Albania        1
1    13.4   11.7         11.3  Czechoslovakia       1
2    11.6   13.4         14.8        Hungary        1
3    13.6   10.7         26.9        Romania        1
4    17.7   10.0         23.0           USSR        1
..    ...    ...          ...            ...      ...
92   50.1   20.2        132.0        Somalia        0
93   44.6   15.8        108.0          Sudan        0
94   31.1    7.3         52.0        Tunisia        2
95   50.5   14.0        106.0       Tanzania        0
96   51.1   13.7         80.0         Zambia        0
[97 rows x 5 columns]
```

Case study

To describe the clusters, we need to study its components:

```
>>> print("Cluster 1:\n", list(data["Country"][(data["Cluster"]==0)]));
    Cluster 1:
    ['Bolivia', 'Mexico', 'Afghanistan', 'Iran', 'Bangladesh', 'Korea', 'Botswana',
     'Gabon', 'Ghana', 'Namibia', 'Sierra_Leone', 'Swaziland', 'Uganda', 'Zaire',
     'Cambodia', 'Nepal', 'Angola', 'Congo', 'Ethiopia', 'Gambia', 'Kenya', 'Malawi',
     'Mozambique', 'Nigeria', 'Somalia', 'Sudan', 'Tanzania', 'Zambia']

>>> print("Cluster 2:\n", list(data["Country"][(data["Cluster"]==1)]));
    Cluster 2:
    ['Albania', 'Czechoslovakia', 'Hungary', 'Romania', 'USSR', 'Ukrainian_SSR',
     'Chile', 'Uruguay', 'Finland', 'France', 'Greece', 'Italy', 'Norway', 'Spain',
     'Switzerland', 'Austria', 'Canada', 'Israel', 'Kuwait', 'China', 'Singapore',
     'Thailand', 'Bulgaria', 'Former_E._Germany', 'Poland', 'Yugoslavia',
     'Byelorussia_SSR', 'Argentina', 'Columbia', 'Venezuela', 'Belgium', 'Denmark',
     'Germany', 'Ireland', 'Netherlands', 'Portugal', 'Sweden', 'U.K.', 'Japan',
     'U.S.A.', 'Bahrain', 'United_Arab_Emirates', 'Hong_Kong', 'Malaysia', 'Sri_Lanka']

>>> print("Cluster 3:\n", list(data["Country"][(data["Cluster"]==2)]));
    Cluster 3:
    ['Ecuador', 'Paraguay', 'Oman', 'Turkey', 'India', 'Mongolia', 'Pakistan',
     'Algeria', 'Egypt', 'Libya', 'Morocco', 'South_Africa', 'Zimbabwe', 'Brazil',
     'Guyana', 'Peru', 'Iraq', 'Jordan', 'Lebanon', 'Saudi_Arabia', 'Indonesia',
     'Philippines', 'Vietnam', 'Tunisia']
```

Case study

The classification of a new observation is not possible through agglomerative algorithms because they do not partition the input space, they just connects some objects during the clustering.

A potential solution could be the use of a supervised machine learning technique, using the output from the agglomerative algorithms as labels, and then classify the new observations accordingly. For instance, we could use the k-nearest neighbour algorithm as follows to classify the following hypothetical countries:

	Birth	Death	InfantDeath
Country A	10	3	5
Country B	29	11	55
Country C	52	25	180

```
>>> from sklearn.neighbors import KNeighborsClassifier
>>> new_data    = pd.DataFrame([[10,3,5], [29, 11, 55], [52, 25, 180]],
                               columns=["Birth", "Death", "InfantDeath"])
>>> new_data_std = pd.DataFrame(fitted.transform(new_data))
>>> knn = KNeighborsClassifier(n_neighbors=1)
>>> knn.fit(X_std, model.labels_)
>>> print(knn.predict(new_data)+1)
    [2 1 3]
```

End