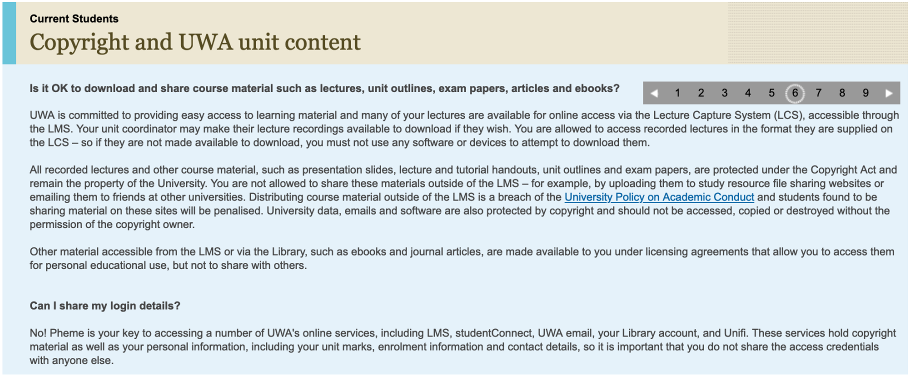


Current Students

Copyright and UWA unit content

Is it OK to download and share course material such as lectures, unit outlines, exam papers, articles and ebooks?

UWA is committed to providing easy access to learning material and many of your lectures are available for online access via the Lecture Capture System (LCS), accessible through the LMS. Your unit coordinator may make their lecture recordings available to download if they wish. You are allowed to access recorded lectures in the format they are supplied on the LCS – so if they are not made available to download, you must not use any software or devices to attempt to download them.

All recorded lectures and other course material, such as presentation slides, lecture and tutorial handouts, unit outlines and exam papers, are protected under the Copyright Act and remain the property of the University. You are not allowed to share these materials outside of the LMS – for example, by uploading them to study resource file sharing websites or emailing them to friends at other universities. Distributing course material outside of the LMS is a breach of the University Policy on Academic Conduct and students found to be sharing material on these sites will be penalised. University data, emails and software are also protected by copyright and should not be accessed, copied or destroyed without the permission of the copyright owner.

Other material accessible from the LMS or via the Library, such as ebooks and journal articles, are made available to you under licensing agreements that allow you to access them for personal educational use, but not to share with others.

Can I share my login details?

No! Pheme is your key to accessing a number of UWA's online services, including LMS, studentConnect, UWA email, your Library account, and Unifi. These services hold copyright material as well as your personal information, including your unit marks, enrolment information and contact details, so it is important that you do not share the access credentials with anyone else.

https://www.student.uwa.edu.au/learning/resources/ace/respect-intellectual-property/copyright-and-uwa-unit-content

CITS5508 Machine Learning

Débora Corrêa (Unit Coordinator and Lecturer)
UWA

2024

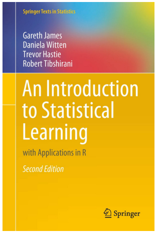
Today

Clustering techniques: Linkage-based Clustering Algorithms.

Understanding Machine Learning (chapter 22)

An Introduction to Statistical Learning (chapter 12)

Applied Unsupervised Learning with Python (chapter 2)



Springer Texts in Statistics

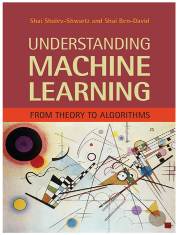
Gareth James
Daniela Witten
Trevor Hastie
Robert Tibshirani

An Introduction to Statistical Learning

with Applications in R

Second Edition

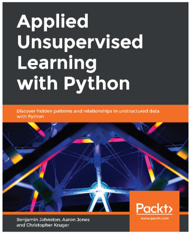
Springer



Shai Shalev-Shwartz and Shai Ben-David

UNDERSTANDING MACHINE LEARNING

FROM THEORY TO ALGORITHMS



Applied Unsupervised Learning with Python

Discover hidden patterns and relationships in unstructured data with Python

Benjamin Johnston, Aaron Jones and Christopher Kruger

Packt>

www.packt.com

Hierarchical Clustering

K-means requires the pre-specification of the number of clusters K.

Hierarchical clustering algorithms do not require a commitment to a particular choice of K.

Hierarchical clustering algorithms are also known as *linkage-based clustering algorithms*.

The results of hierarchical clustering algorithms are generally presented in a `dendrogram`, which are tree-based representations of the instances.

Hierarchical Clustering

The idea of hierarchical clustering is to build a hierarchy of clusters by following a sequence of rounds starting from the trivial clustering:

- Each data point is a single-point cluster, then repeatedly merge clusters as going up in the hierarchy. This is a "bottom-up" approach also known as `agglomerative hierarchical clustering`; or
- All instances start as one cluster, then repeatedly split clusters as going down in the hierarchy. This is a "top-down" approach also known as `divisive hierarchical clustering`.

Agglomerative Hierarchical Clustering

Each instance starts as a single-point cluster.

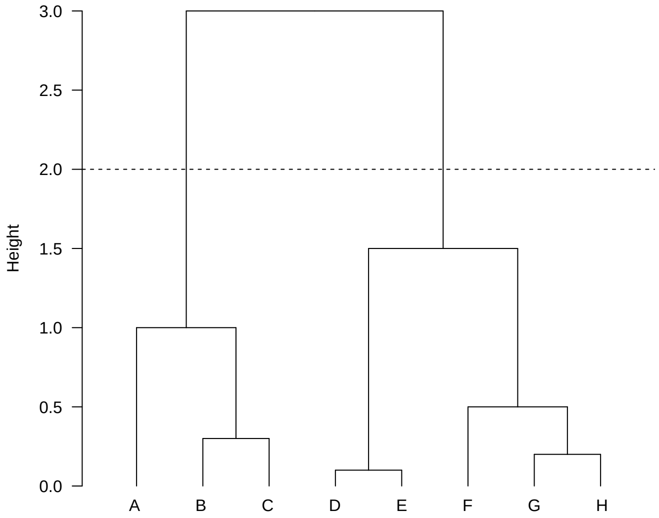
Then, repeatedly, merge the “closest” clusters of the previous clustering.

The number of clusters decreases after each iteration. The algorithm will eventually find the trivial clustering in which all instances share a single cluster.

We need to define two criteria:

- how to measure (or define) the distance between two instances; and
- how to measure (or define) the distance between two clusters. This is *linkage* criterion.

Hierarchical Clustering Example



Interpreting the Dendrogram

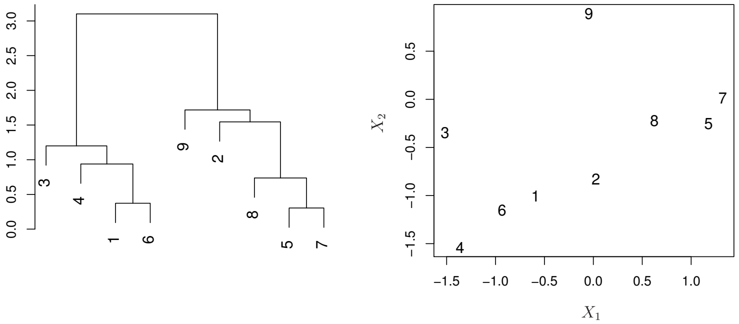
Starting from the leaves, the clusters are combined up to the trunk (where all instances belong to the same cluster).

Each leaf at the bottom of the dendrogram represents an instance. Similar leaves are joined to create a node. Then, similar nodes are joined until all instances share the same cluster.

Instances joining at the bottom of the dendrogram tend to be more similar than the instances joining close to the top of the tree.

More specifically: for any two instances, we find the point in the dendrogram where the branches containing those two instances are first joined. The height of this fusion (measured in the vertical axis) indicates how different the two instances are.

Interpreting the Dendrogram

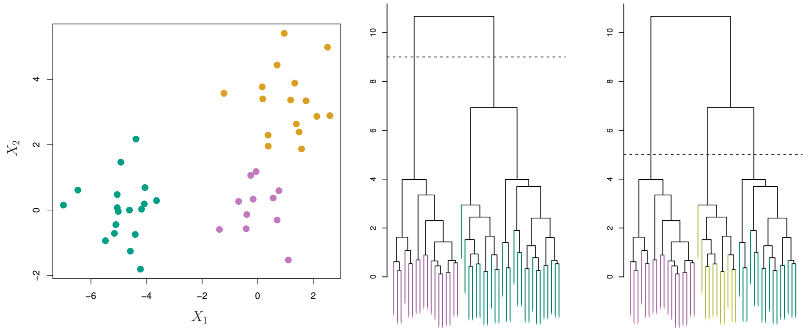


Interpretation of the dendrogram given the nine instances on the right.

We conclude about the similarity between two instances based on the location on the *vertical* axes where branches with the two instances are first merged.

The Number of Clusters

A cut in the dendrogram at a specific height will result in distinct sets of instances that can be interpreted as clusters.



Right: Forty-five observations generated in two-dimensional space. Left:Dendrograms obtained from hierarchically clustering the data on the right.

The Hierarchical Clustering Algorithm

We need to define the *dissimilarity* measure between two instances. A popular choice is the Euclidean distance.

Initial condition (the bottom of the dendrogram): each of the m instances is treated as a cluster.

Then, the two clusters that are most similar to each other are then merged, resulting in $m - 1$ clusters.

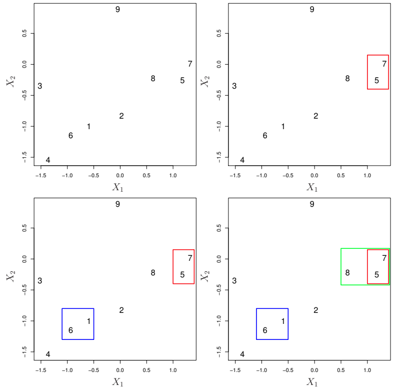
In the next iteration, the next two clusters that are most similar to each other are merged, resulting in $m - 2$ clusters.

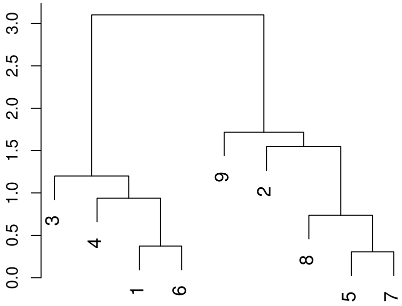
This process is repeated until all instances are part of the same cluster.

The Hierarchical Clustering Algorithm

1. Begin with m instances as their own clusters. Define a dissimilarity measure for all the $\binom{m}{2} = m(m-1)/2$ pairwise dissimilarities.
2. For $i = m, m-1, ..., 2$:
    a. From all pairwise inter-cluster dissimilarities among the i clusters, identify the pair of clusters that are least dissimilar (that is, most similar). Merge these two clusters. Such dissimilarity between these two clusters determines the height in the dendrogram at which the merging should be placed.
    b. Compute the new pairwise inter-cluster dissimilarities among the $i-1$ remaining clusters.

The Hierarchical Clustering Algorithm





An illustration for the first few steps of the hierarchical clustering algorithm, using nine instances and with complete linkage and Euclidean distance.

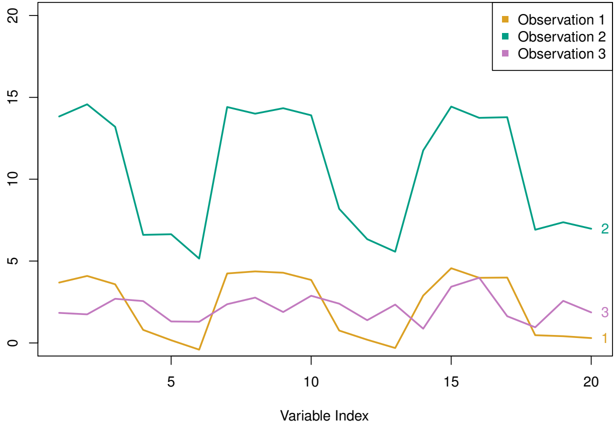
Choice of Dissimilarity Measure

Some options include:

- Euclidean distance.
- Correlation between instances (emphasis is on similar shape, not magnitude)
- Application-relevant measures.

Euclidean and Correlation-based Distance

Correlation-based distance focuses on the shapes of the instance profiles.



Three observations with measurements on 20 variables.

Linkage

We also need to define a concept of dissimilarity between two clusters, where they can contain multiple instances.

That is, the concept of dissimilarity between a pair of instances needs to be extended to a pair of clusters (or groups of instances).

This is where the notion of `linkage` is incorporated.

Four of the most popular types of linkage: complete, single, average and centroid.

The resulting dendrogram will typically depend on the type of linkage used, as well as on the choice of dissimilarity measure.

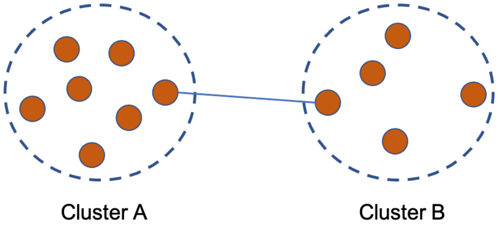
Average and complete tend to be the preferred types of linkage.

Single Linkage

Minimal inter-cluster dissimilarity.

Compute all pairwise dissimilarities between the instances in cluster A and the instances in cluster B. Use the *smallest* of these dissimilarities as the dissimilarity between the two clusters.

$$D(A, B) \overset{\text{def}}{=} min\{d(x, y) : x \in A, y \in B\}$$

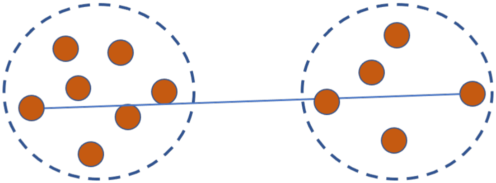


Complete Linkage

Maximal inter-cluster dissimilarity.

Compute all pairwise dissimilarities between the instances in cluster A and the instances in cluster B. Use the *largest* of these dissimilarities as the dissimilarity between the two clusters.

$$D(A, B) \overset{\text{def}}{=} max\{d(x, y) : x \in A, y \in B\}$$

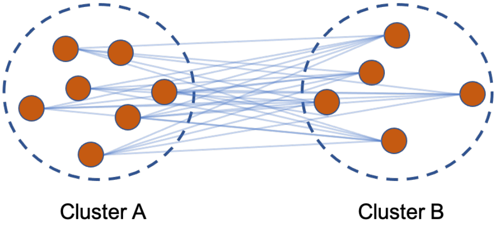


Average Linkage

Mean inter-cluster dissimilarity.

Compute all pairwise dissimilarities between the instances in cluster A and the instances in cluster B. Use the *average* of these dissimilarities as the dissimilarity between the two clusters.

$$D(A, B) \overset{\text{def}}{=} \frac{1}{|A||B|} \sum_{x \in A, y \in B} d(x, y)$$



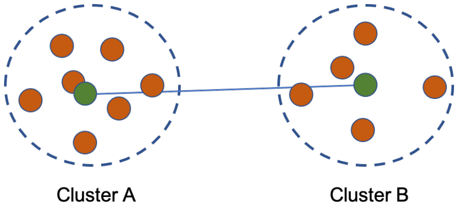
Centroid Linkage

Centroid inter-cluster dissimilarity.

Compute the centroid for cluster A (a mean feature vector of length n) and the centroid for cluster B. Use the dissimilarity between the two centroids as the dissimilarity between the two clusters. Note: n is the number of features.

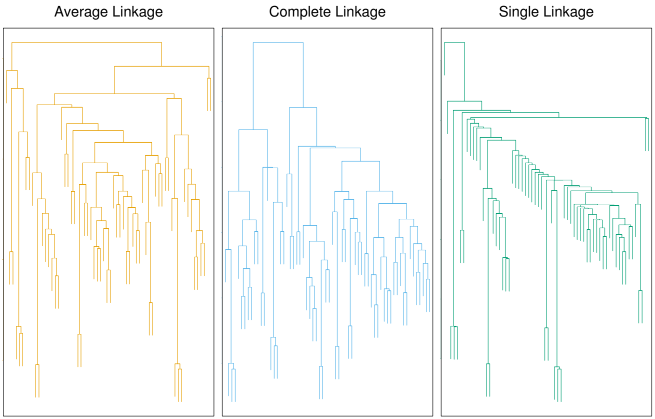
$$D(A, B) \overset{\text{def}}{=} d(\mu_A, \mu_B)$$

where μ_A and μ_B are the centroids of clusters A and B, respectively.



Comparing the Dendrograms

Illustration of different types of linkage.

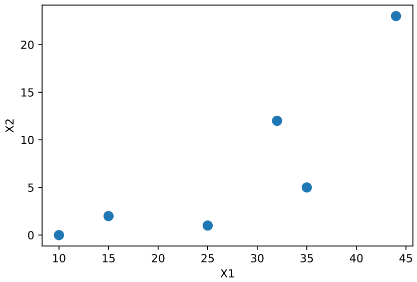


Average, complete and single linkage on an example data.

Example

Let's consider an example with six instances and two features.

	X_1	X_2
1	35	5
2	10	0
3	32	12
4	44	23
5	15	2
6	25	1



Example

Distance matrix given by the Euclidean distance:

	1	2	3	4	5	6
1	0.00	25.50	7.62	20.12	20.22	10.77
2	25.50	0.00	25.06	41.05	**5.39**	15.03
3	7.62	25.06	0.00	16.28	19.72	13.04
4	20.12	41.05	16.28	0.00	35.81	29.07
5	20.22	**5.39**	19.72	35.81	0.00	10.05
6	10.77	15.03	13.04	29.07	10.05	0.00

Example

Single Linkage: minimal distance between clusters.

The shortest distance in our distance matrix is 5.39 associated with instances 2 and 5.

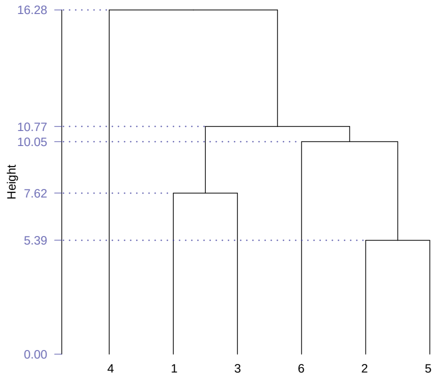
Therefore, we merge these two clusters, creating the new cluster 2-5, and compute the distance between the cluster 2-5 with the other clusters. For example:

- 1 to 2-5 = min(1→2,1→5) = min(25.50,20.22) = 20.22
- 3 to 2-5 = min(3→2,3→5) = min(25.06,19.72) = 19.72

	1	2-5	3	4	6
1	0	20.22	**7.62**	20.12	10.77
2-5	20.22	0	19.72	35.81	10.05
3	**7.62**	19.72	0	16.28	13.04
4	20.12	35.81	16.28	0	29.07
6	10.77	10.05	13.04	29.07	0

Example

Now the shortest distance is 7.62, merging clusters 1 and 3, creating the new cluster 1-3, and the process repeats.



	1-3	2-5	4	6
1-3	0	19.72	16.28	10.77
2-5	19.72	0	35.81	**10.05**
4	16.28	35.81	0	29.07
6	10.77	**10.05**	29.07	0

Stopping Criteria

To turn a dendrogram into a partition of the space (a clustering), we need to define a stopping criterion. Two possibilities are:

- Fix the number of clusters. Stop merging clusters when the desired number of clusters is achieved;
- Fix a distance threshold. Stop merging when all the between-clusters distances are larger than this threshold.

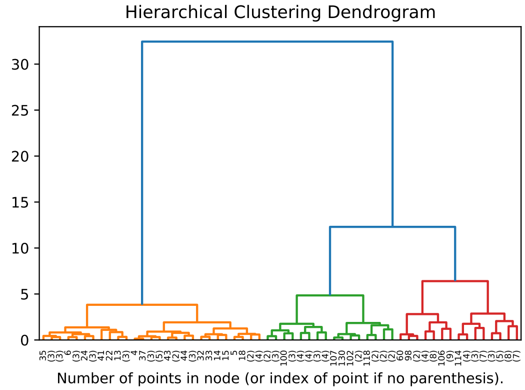
Hierarchical Clustering with Scikit-learn

Scikit-learn implements hierarchical clustering in the class `AgglomerativeClustering`. The class `dendrogram` can be used to plot the dendrogram.

```
from sklearn.datasets import load_iris
from sklearn.cluster import AgglomerativeClustering
from scipy.cluster.hierarchy import dendrogram

iris = load_iris()
X = iris.data

# setting distance_threshold=0 ensures we compute the full tree.
# model = AgglomerativeClustering(distance_threshold=0, n_clusters=None)

model = AgglomerativeClustering(n_clusters=3, compute_distances= True)

model = model.fit(X)

model.labels_
>>> array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
          ...
          2, 0, 0, 2, 2, 2, 0, 2, 2, 2, 0, 2, 2, 2, 0, 2, 2, 0])
```

Hierarchical clustering with Scikit-learn

```
plt.title("Hierarchical Clustering Dendrogram")

# plot the top five levels of the dendrogram
plot_dendrogram(model, truncate_mode="level", p=5, color_threshold=6.5)
plt.xlabel("Number of points in node (or index of point if no parenthesis).")
plt.show()
```



Practical Aspects

Different decisions may lead to different results:

- Should we standardise the data?
- What to use as the dissimilarity measure between a pair of instances?
- What to use as the dissimilarity measure between a pair of clusters (that is, the type of linkage)?
- Where to cut the dendrogram to obtain the clusters?

There is no unique or single right answer. Different choices expose interesting aspects of the data and should be considered.

Look for the most useful or interpretable solutions according to the application. Always consider the type of data being clustered and the scientific question addressed.

Example: Clustering for Shopping-related Data

An online retailer wants to cluster shoppers based on their past shopping histories. That is, the goal is to identify subgroups of similar shoppers.

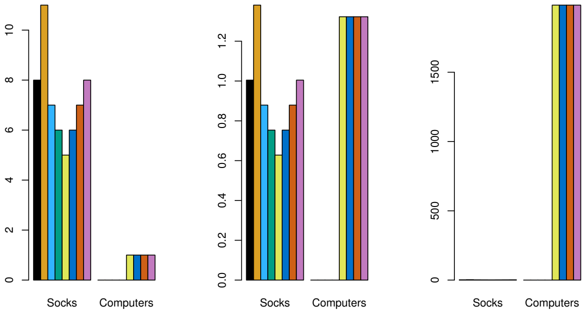
Suppose in our data matrix, each row represents a shopper, and each column represents the number of times a shopper has purchased a specific item.

Euclidean distance as distance measure: infrequent shoppers may be clustered together.

Correlation-based distance: shoppers with similar preferences will be clustered together (even if some of these shoppers are higher-volume shoppers).

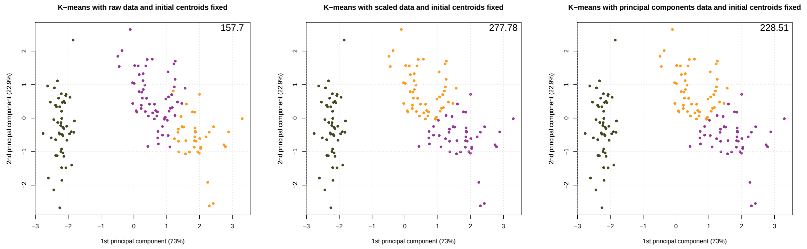
Example: Clustering for Shopping-related Sata

Consider whether or not the variables should be scaled before the dissimilarity between the instances is computed. Shall we give equal importance to items, even if some items may be purchased more frequently than others?



The decision to scale or not the variables before computing the dissimilarity measures will depend on the application at hand.

K-means Centroids as the Output of the Hierarchical Clustering



In this particular example, we know the labels. Therefore, we can compute the “accuracy” of these three solutions, leading to 89.33%, 83.33% and 83.33%, respectively.

For next week

Work through Assignment 3 and attend a supervised lab. The Unit Coordinator or a casual Teaching Assistant will be there.

Read Chapters on Neural Networks in the textbook.