Machine Learning

Hierarchical Clustering

Ayşe Ceren Çiçek

Hierarchical clustering is **an algorithm that groups similar objects into groups called clusters**. It is an alternative approach to k-means clustering for identifying groups. The endpoint is a set of clusters, where each cluster is distinct from the other cluster, and the objects within each cluster are broadly similar to each other.

- It is an unsupervised machine learning algorithm.
- The hierarchical clustering does not require us to pre-specify the number of clusters to be generated as is required by the k-means approach.
- It has a tree-based representation called **dendrogram** which is a diagram that shows the hierarchical relationship between objects.

Hierarchical clustering can be divided into two main types: **agglomerative** and **divisive**.

A) Agglomerative Clustering

It is a bottom-up approach. In the beginning, each object is initially considered as a single-element cluster. At each step of the algorithm, the two clusters that are the most similar are combined into a new bigger cluster. This procedure is iterated until all points are member of just one single big cluster.

This process has a **O(N^3) time complexity** and a **O(N^2) memory complexity** that makes it *not tractable for large datasets.*

How it works:

- Make each data point a single-point cluster
- Take the two closest *data points* and make them one cluster
- Take the two closest *clusters* and make them one cluster
- Repeat the previous step until there is only one cluster

B) Divisive Clustering

It is a top-down approach. It begins with the root, in which all objects are included in a single cluster. At each step of iteration, the most heterogeneous cluster is divided into two. The process is iterated until all objects are in their own cluster.

This process requires at each iteration to search for the best split, implying a **O(2N) time complexity** that has to be tackled with some heuristics. *Divisive hierarchical clustering is good at identifying large clusters.*

Importing libraries

```
library(cluster)
```

Dataset: https://www.kaggle.com/shwetabh123/mall-customers

The columns are as follows:

- CustomerID: It is the unique ID given to a customer
- Gender: Gender of the customer
- Age: The age of the customer
- Annual Income(k$): It is the annual income of the customer
- Spending Score: It is the score(out of 100) given to a customer by the mall authorities, based on the money spent and the behavior of the customer.

Loading dataset

```
dataset = read.csv('dataset.csv')
head(dataset, n=5)
```

```
##   CustomerID  Genre Age Annual.Income..k.. Spending.Score..1.100.
## 1          1   Male  19                 15                     39
## 2          2   Male  21                 15                     81
## 3          3 Female  20                 16                      6
## 4          4 Female  23                 16                     77
## 5          5 Female  31                 17                     40
```

We will use annual income and spending score to cluster customers.

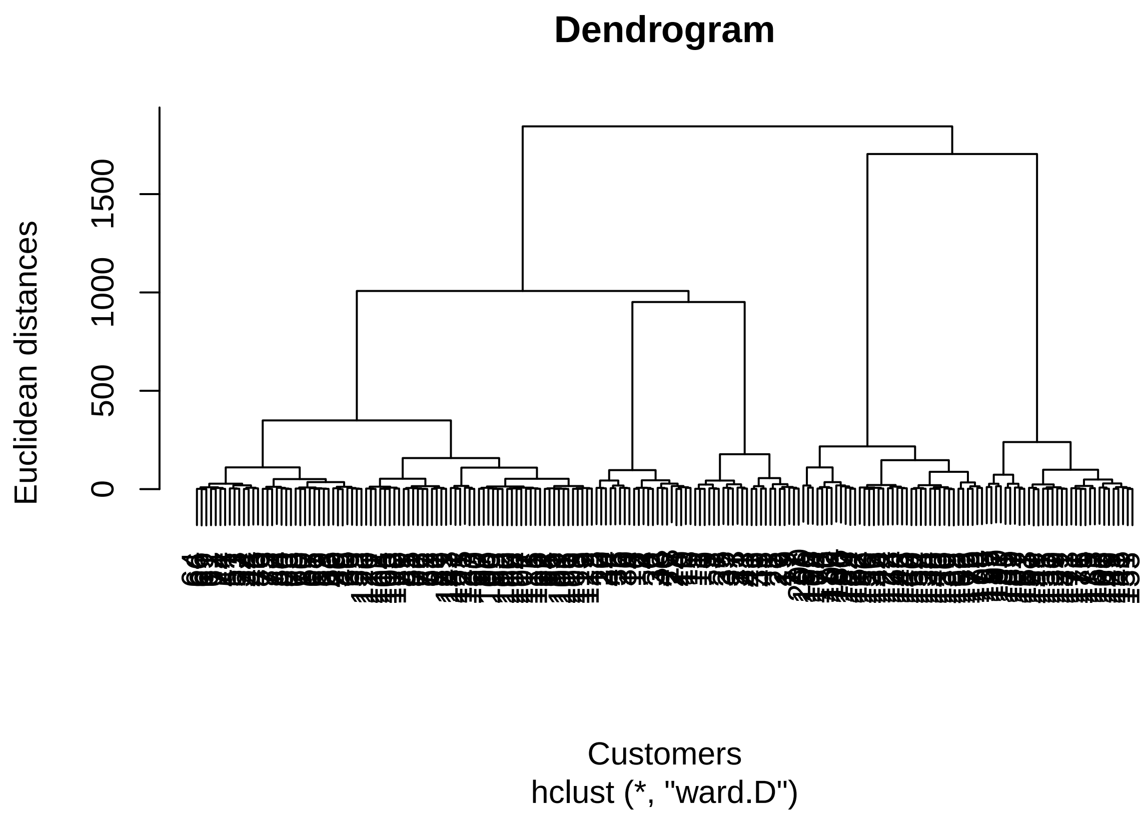
```
X <- dataset[4:5]
head(X, n=5)
```

```
##   Annual.Income..k.. Spending.Score..1.100.
## 1                 15                     39
## 2                 15                     81
## 3                 16                      6
## 4                 16                     77
## 5                 17                     40
```

Dendrogram

A dendrogram is a tree-like chart that shows the sequences of merges or splits of clusters. We will use it to find the optimal number of clusters.

```r
dendrogram <- hclust(dist(X, method = 'euclidean'), method = 'ward.D')
plot(dendrogram, main = 'Dendrogram', xlab = 'Customers', ylab = 'Euclidean distances')
```



Optimal number of clusters is 5.

Apply hierarchical clustering

```r
hc <- hclust(dist(X, method = 'euclidean'), method = 'ward.D')
```

Cutree method cuts a dendrogram tree into several groups by specifying the desired number of clusters k(s), or cut height(s).

```r
y_hc <- cutree(hc, 5)
y_hc
```

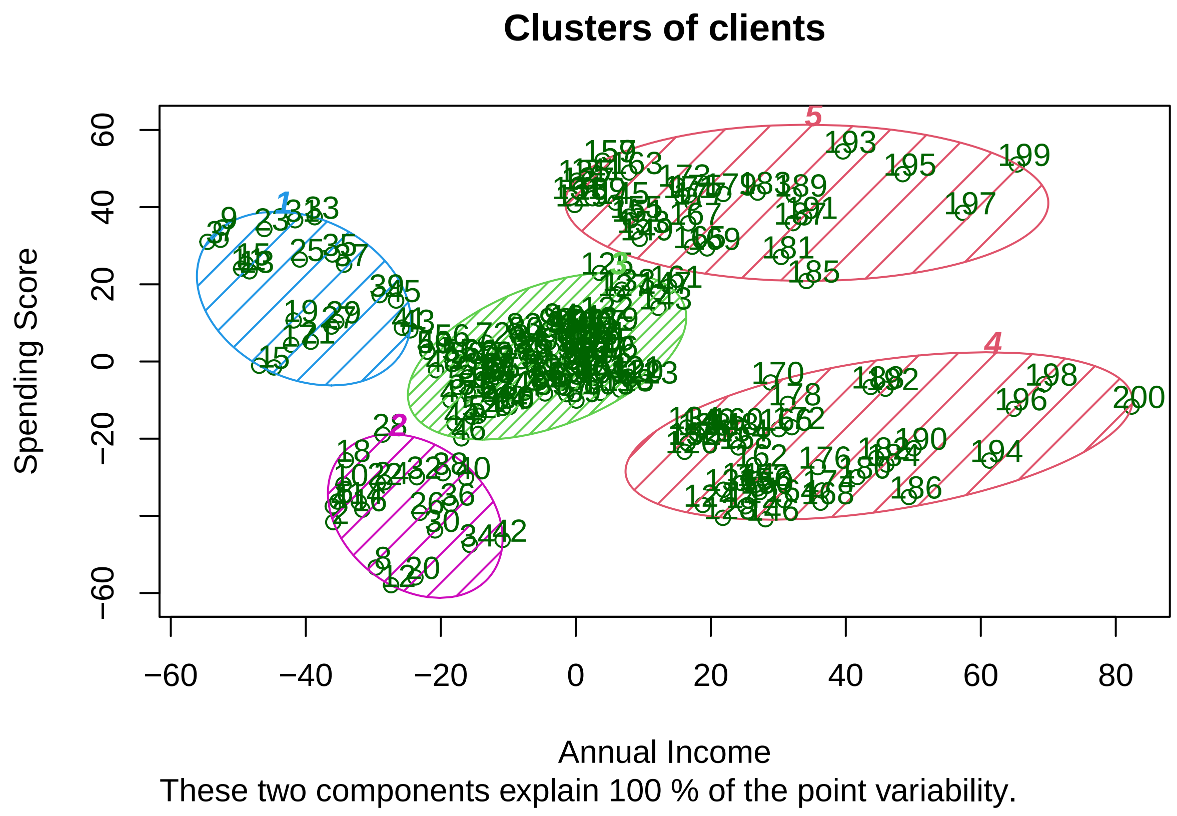
```
##   [1] 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1
##  [38] 2 1 2 1 2 1 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##  [75] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [112] 3 3 3 3 3 3 3 3 3 3 3 4 3 4 3 4 5 4 5 4 3 4 5 4 5 4 5 4 5 4 3 4 5 4 3 4
## [149] 5 4 5 4 5 4 5 4 5 4 5 4 3 4 5 4 5 4 5 4 5 4 5 4 5 4 5 4 5 4 5 4 5 4 5 4 5
## [186] 4 5 4 5 4 5 4 5 4 5 4 5 4 5 4
```

We can see the clusters above.

Visualize the clusters

With clusplot function we can draw a 2 dimensional clustering plot with our clusters.

```r
clusplot(X, clus = y_hc, lines = 0, shade = TRUE, color = TRUE, labels = 2, plotchar = FALSE, span = TRU
         main = paste("Clusters of clients"), xlab = "Annual Income", ylab = "Spending Score")
```


Clusters of clients
Spending Score
Annual Income
These two components explain 100 % of the point variability.