GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN

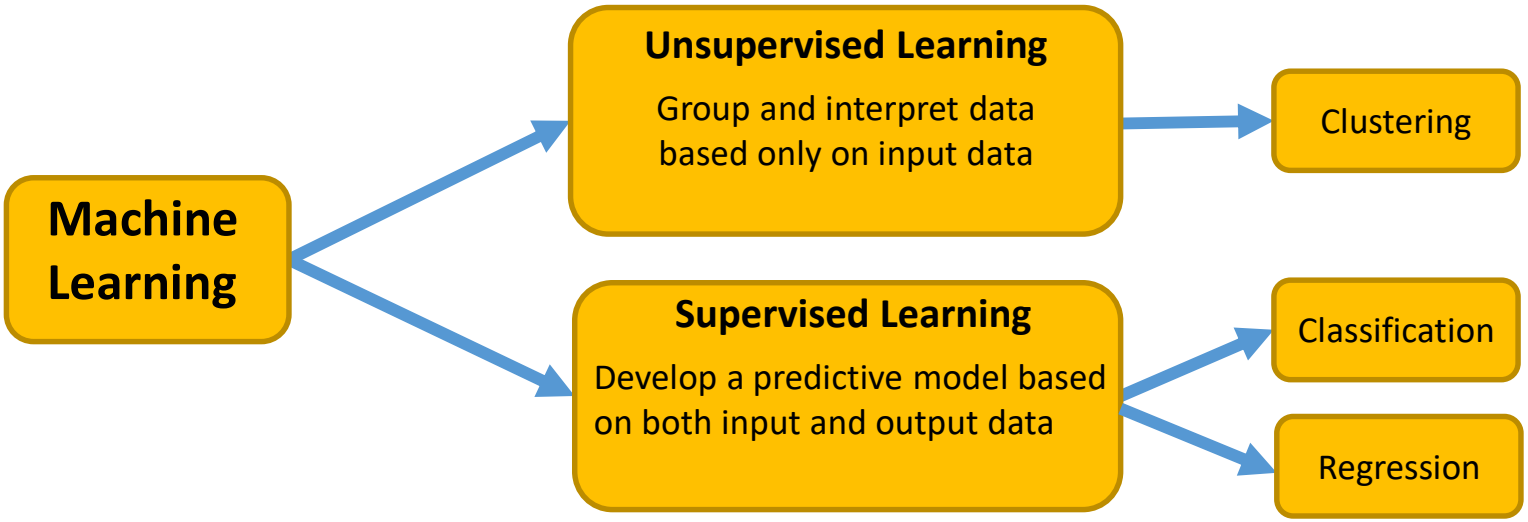
Applied Machine Learning with R

Felix Heinrich

Breeding Informatics Group
Department of Animal Science
Georg-August University Göttingen

Algorithms

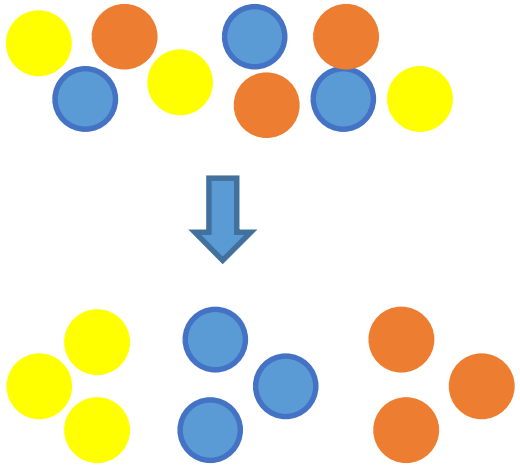
- The success of machine learning systems also depends on the algorithms.
- The algorithms control the search to find and build the knowledge structures.
- The learning algorithms should extract useful information from training examples.



Clustering

- A cluster is a group of related objects

- Clustering:
  - Identification of natural groups/subsets of objects in a data set



Data clustering

A good clustering method detects high quality clusters in which:

- The **homogenity** is very high: **intra-cluster** similarity
- The **separation** is very high: **inter-cluster** similarity

- The quality of a clustering result → similarity measure used

Aim of clustering

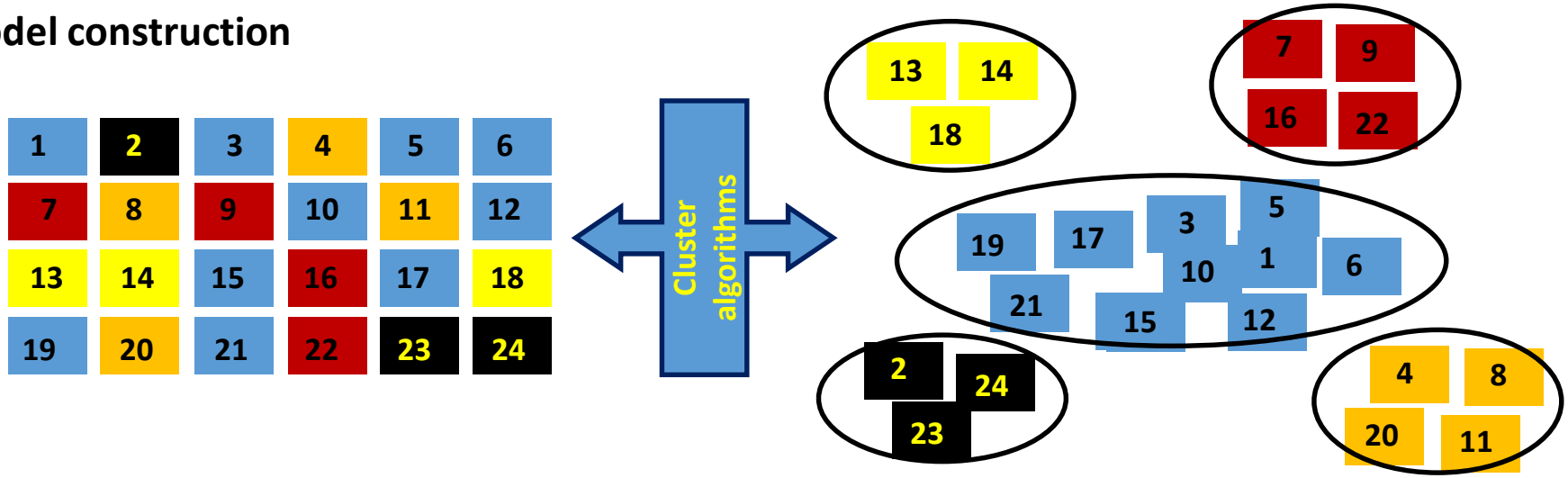
Clustering is a function that maximizes similarity between objects within a cluster and minimizes similarity between objects in different clusters.

Data clustering

- Identify relationships and patterns in a set of data
- Identify similar groups between genes or proteins
- Get more insights in underlying data structure
- Deal with numerical measures of the data
  - Distance metrics, d
    - $d(a,b) \geq 0$
    - $d(a,a) = 0$
    - $d(a,b) = 0$ if and only if $a = b$
    - $d(a,b) = d(b,a)$ → symmetry
    - $d(a,c) \leq d(a,b) + d(b,c)$ → triangle inequality

Clustering process:

1. **Model construction**



The process of putting similar data together

- No training data
- A notion is needed (e.g. similarity or distance)

Clustering

Clustering
Unknown number of groups
No prior knowledge
Used to explore data
A form of **unsupervised** learning

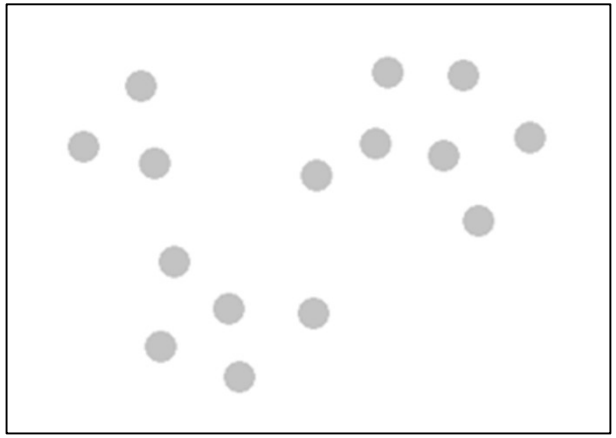
You can compute it without the need of knowing the correct solution

k-means clustering

- **k-means** is, without doubt, the most popular clustering method
- **k-means** is one of the simplest and most widely used clustering algorithms
- **k-means** aims to find the set of ***k*** clusters such that
  - every data point is assigned to the closest center,
  - the sum of the distances of all such assignments is minimized

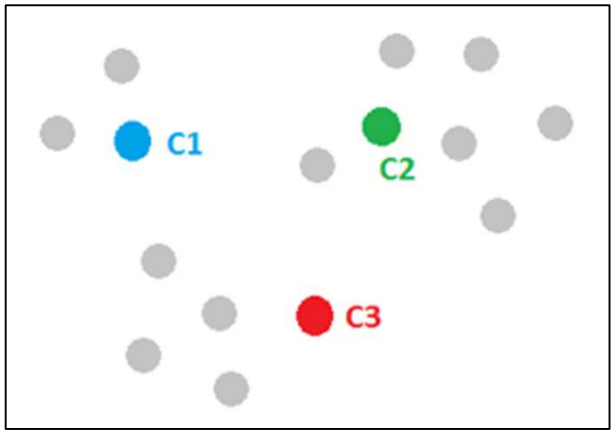
Example: Assign the gray points into three clusters.

Input data: not clustered



Step one: Initialize cluster centers

- Randomly pick three points C1, C2, and C3
- Label them with blue, green and red color

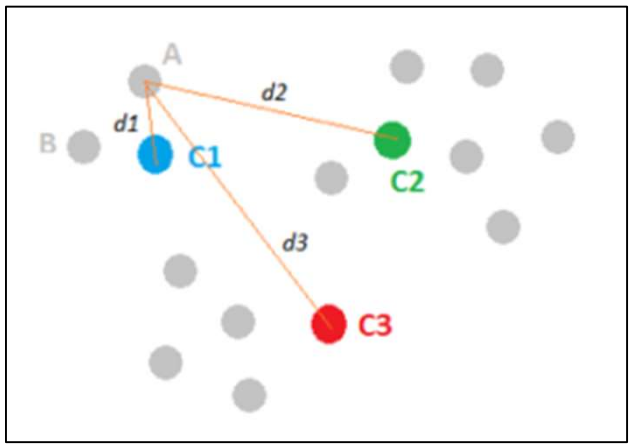


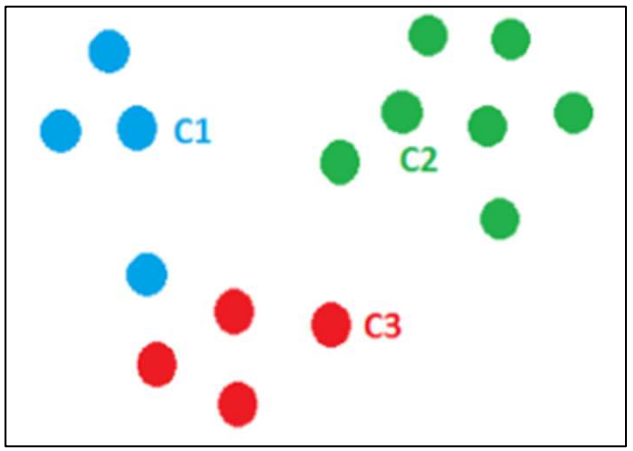
k-means clustering

Example: Assign the gray points into three clusters.

Step two: Assign observations to the closest cluster center

– Label them with **blue**, **green** and **red** color



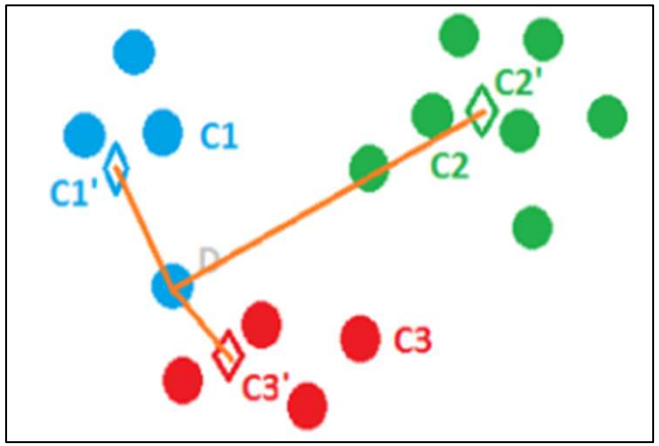


Assign each point to the clusters based on the minimum distance to the cluster center

k-means clustering

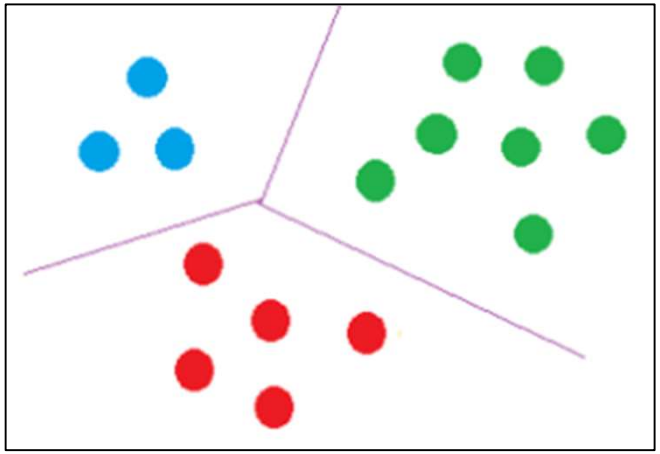
Example: Assign the gray points into three clusters.

Step three: Revise cluster centers as mean of assigned observations



Update the cluster centers

Step four: Repeat step 2 and 3 until convergence



k-means clustering

Example:

Import the data file called **"plantData.csv"** into the variable **mydata**

```
> mydata= read.table("plantData.csv", header=TRUE, sep="," , stringsAsFactors = TRUE)
 # Check the dimensions of the data
 # View statistical  summary of dataset
 # View the complete data
```

Process the dataset

- Divide **mydata** in two data frames as **df1** and **df2**
  - **df1** contains the class attribute "Species"
  - **df2** contains the remaining information
- Write a function to normalize the values in **df2** using **min-max normalization**
- Apply **k-means** clustering algorithm (function in R: **kmeans**)
- Verify results of clustering by plotting them

k-means clustering

Example:

Process the dataset

- Divide **mydata** in two data frames as **df1** and **df2**

```
> df1=mydata[, "Species"]
> df2=mydata[, c(1,2,3,4)]
```

- Normalize the values in **df2** between 0 and 1 using our own function

```
> normalize= function(x){
+             normalizedValues= (x-min(x))/(max(x)-min(x))
+              return (normalizedValues)
+}
# you can normalize each attribute manually or use a  for loop
>df2[, "Sepal.Length"]=normalize(df2[, "Sepal.Length"])  # repeat it for each attribute
>for(i in 1:ncol(df2)){
+             df2[, i] = normalize(df2[,i])
+}
```

k-means clustering

Example:

Process the dataset

- Apply **k-means** clustering algorithm (?kmeans for help)

```
> clusterResult= kmeans(df2,centers=3) #apply k-means algorithm with k=3
> clusterResult$size # gives number of records in each cluster
```

- Verify results of clustering by plotting them

k-means clustering

Example:

Process the dataset

– Verify results of clustering by plotting them with plot()

```
>par(mfrow=c(1,2)) # multiple plots in same window

# Plot to see how Sepal.Length and Sepal.Width data points have been distributed in clusters
>plot(df2[c(1,2)], col= clusterResult$cluster)

# Plot to see how Sepal.Length and Sepal.Width data points are distributed originally
>plot(df2[c(1,2)], col=df1)

>plot(df2[c(3,4)], col= clusterResult$cluster)
>plot(df2[c(3,4)], col=df1) # Plot to see how Petal.Length and Petal.Width data points have been
distributed originally as per "class" attribute in dataset
```