# Lecture3

John Salmon

2025-02-03

## Lecture 3: Clustering

### Clustering Motivation

- Marketing
- Medicine
- Economics

Clustering is often used in exploratory data analysis or as a starting point of further study.

**Basic Principles** In general it refers to a broad set of techniques for finding subgroups or clusters in a data set. - Observations within each group are quite similar to each other - Observations in different groups are quite different from each other

Clustering depends on the concept of "similarity" (or "Dissimilarity)

2 Popular clustering methods - k-means clustering - hierarchical clustering Specificity in your method is better Hierarchical clustering is typically more flexible.

function for clustering in R *clusGap* from *cluster* , and *ElemStatLearn* for the data set

Working with Iris data set for examples again. Iris is dimension $p = 4$

Euclidean distance $d(x_i, x_j) = (\sum_{k=1}^{p}(x_{ik} - x_{jk})^2)^{\frac{1}{2}}$ Distance between two of the same vector $= 0$ Distance between two opposite vectors $= 2||v||$ if 2 vectors V & W where $W = -V$

3 rules for distance $d(u, w)+d(w, z) \geq d(u, z)$ Triangular inequality $d(u, w) = 0 \iff u = w$ $d(u, w) = d(w, u)$

dissimilarity between $x_i$and$x_j$ is defined as $d_{ij} = d^2(x_i, x_j)$ smaller $d_{ij}$ means $x_i$and$x_j$ are more similar and less dissimilar.

it is much easier to optimize square functions than square root functions because ttaking the derivative of a square root functioin explodes to - as f -> 0.

Using squared euclidean distance is better than using euclideian distance.

```
x1 = c(5.1,3.5,1.3,0.2)
x2 = c(4.9,3.0,1.4,0.2)
x3 = c(4.7,3.2,1.3,0.2)

sqrt(sum((x1-x2)^2))
```

```
## [1] 0.5477226
```

```r
sqrt(sum((x1-x3)^2))
```

```
## [1] 0.5
```

K-means clustering divides observations int K disjoint clusters and assigns each observation to a cluster. - When $K = 2$ each observations will be assigned to one of 2 disjoint clusters - Let $k = 1$ or 2 denote the cluster membership of an observation.

Clustering is a mapping C that assigns the ith observation $x_i$ to Cluster $k$ This mapping is many-to-one it is never many-to-one because the clusters are disjoint

Modern Deep learning classification is one-to-many $BUT$ probabilisticlly

*Clustering is just label assignment*

The loss function for K-means seeks to minimize the dissimilarity betweenn points in the same cluster. In the extreme case where all observations are the same you would hope there would only be 1 cluster.

The number of combinations for N observations disjoint clustering is $N!$(n factorial $n * (n-1) * (n-2) * ... * (n-n+1)$) Clustering is a combinatorial complexity which is NP hard.

1. Ask the correct questions
2. Find approximate answers The paradigm has shifted as processing and time is limited.

K-means algorithm 1. Randomly assign a number 1 to k to each observation 2. iterate the following steps 2. (a) for each cluster compute the centroid 2. (b) assign each observation to the cluster whose centroid is closest.

When cluster assignment stops changing we say the algorithm "converges" (perhaps to a local minimum of the loss function) The algorithm will always converge to a local minimum, but it will not always converge to the global minimum Different initial configurations for the Algorithm may give different clustering results when it converges. The algorithm needs to run under multiple initial configurations to help get closer to the global minimum of the loss function by choosing a best initial configuration.

Other dissimilarity Measures Euclidean distance is not invariant to vector length

- Correlation Distance $1 - corr(x_i, x_j)$ where corr is the Pearson correlation between the p samples. Its 1 minus correlation because that way the value can never be negative, there is no negative distance. Pearson correlation is invariant when it comes to the length of a vector. Pearson Correlation is equivalent to the Cosine of the angle between two vectors.
- Manhattan distance $\sum_{k=1}^{p} \mid x_{ik} - x_{jk} \mid$ Frequently used in computer science to convert binary strings.

Qualitative Features - For ordinal variables, map them using $y- > \tilde{y} = [rank(y) - 2/M]/M$ - For Nominal Variables (such as gender): for values a and b in the range of the variable, set $d(a, b) = 0$ when $a = b$ and $d(a, b) = 1$ for when $a \neq b$

True Number of Clusters $K_0$ is unknown and needs to be estimated.

- the "gap statistic" can be used to estimate the number of clusters. It's based on differences between within-cluster sums of squares, and is implemented by the function `clusGap` of the library `cluster`. The gap statistic does not estimate $K_0$ for each dataset. Instead a set of assumptions should be satisfied to ensure its accuracy.

**K means syntax** The command is very simple. ?kmeans for more info. - X: data object as matrix - centers: number of clusters to form, or a matrix containing centroids - nstart: number of random, initial configurations of cluster memberships. - algorithm: the algorithm used to minimize the loss function - iter.max the maximum number of iterations run by the algorithm

the kmeans command returns: - Cluster: a vector of integers for the value $k_i$ of whose $ith$ entry indicates that the $ith$ observation is assigned to Cluster $k_i$, where K is the number of clusters to form - Centers: a matrix of cluster centers, whose $jth$ row contains the coordinates of the centroid of Cluster j - tot.withns: the value of the loss function when the algorithm converges. Total within-cluster sum of squares.

Notes: - Kmeans cannot be applied to non-numerical data (i.e. character string, ordinal or nominal) These must be converted. - Even with the same number of clusters different nstart or iter,max values may give different clustering results.

```
set.seed(2)
x = matrix(rnorm(50*2), ncol=2)
x[1:25,1] = x[1:25, 1] + 3
x[1:25, 2] = x[1:25, 2] -4
#create a dataset split into two clusters by changing mean vectors of 25 obs. at a time.
```

**Example 1** The family of Gaussian distributions are closed under both scaling, and translation. $Y = \sigma X + \mu \sim N(\mu, \sigma)$

$X \sim N(2, 1)$, $Y \sim N(0, 4)$

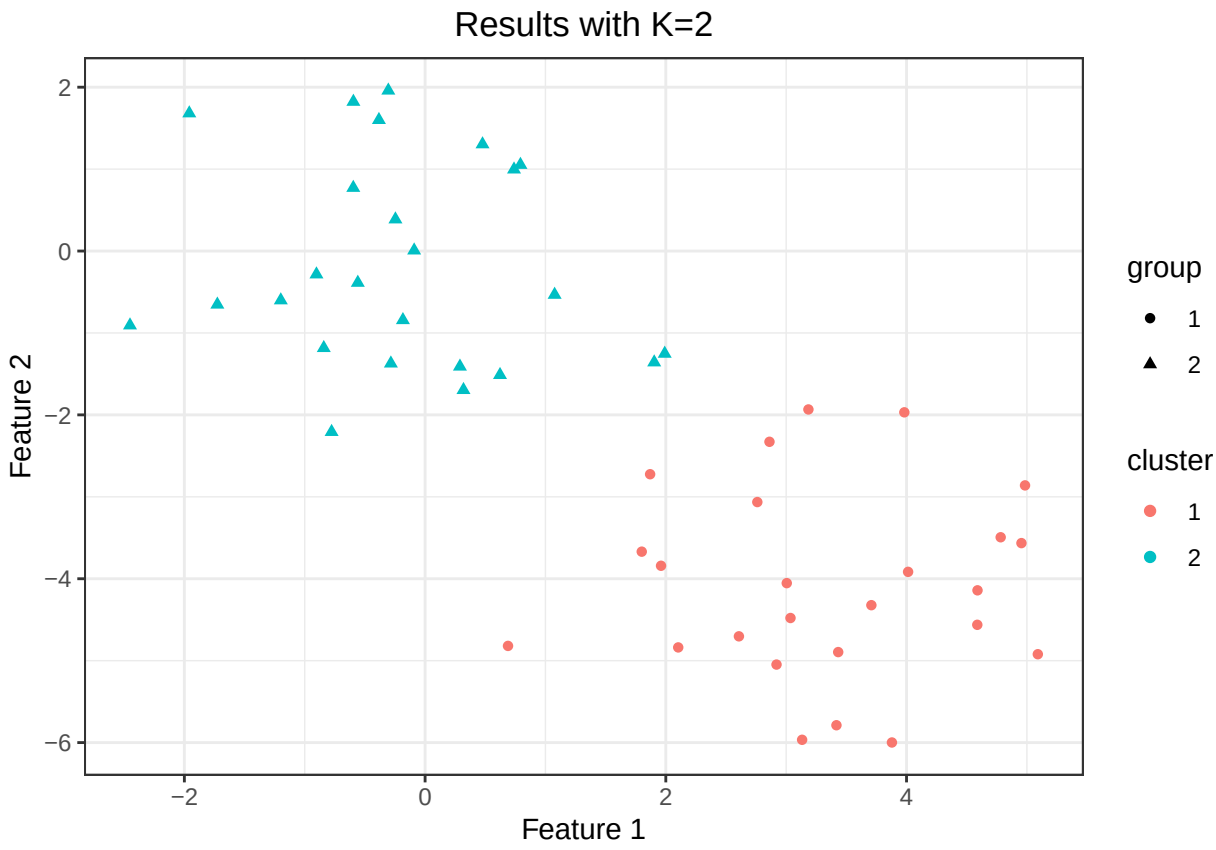
If x and y are independent then $Z = (X, Y)$ must be a dist f dim 2

with a set of 50 observations(rows), with 2 columns, there are 50 mutually independent observations, meaning if you pick any 2 they will be independent. Independence implies that covariance is 0. 5 params to determine a 2-D normal distribution mean & variance of each variable, and covariance of both. 1-D you only need 2. Mean, variance.

```
km.out=kmeans(x,2,nstart=20)
km.out
```

```
## K-means clustering with 2 clusters of sizes 25, 25
##
## Cluster means:
##         [,1]       [,2]
## 1  3.3339737 -4.0761910
## 2 -0.1956978 -0.1848774
##
## Clustering vector:
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2
## [39] 2 2 2 2 2 2 2 2 2 2 2 2
##
## Within cluster sum of squares by cluster:
## [1] 63.20595 65.40068
##  (between_SS / total_SS =  72.8 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

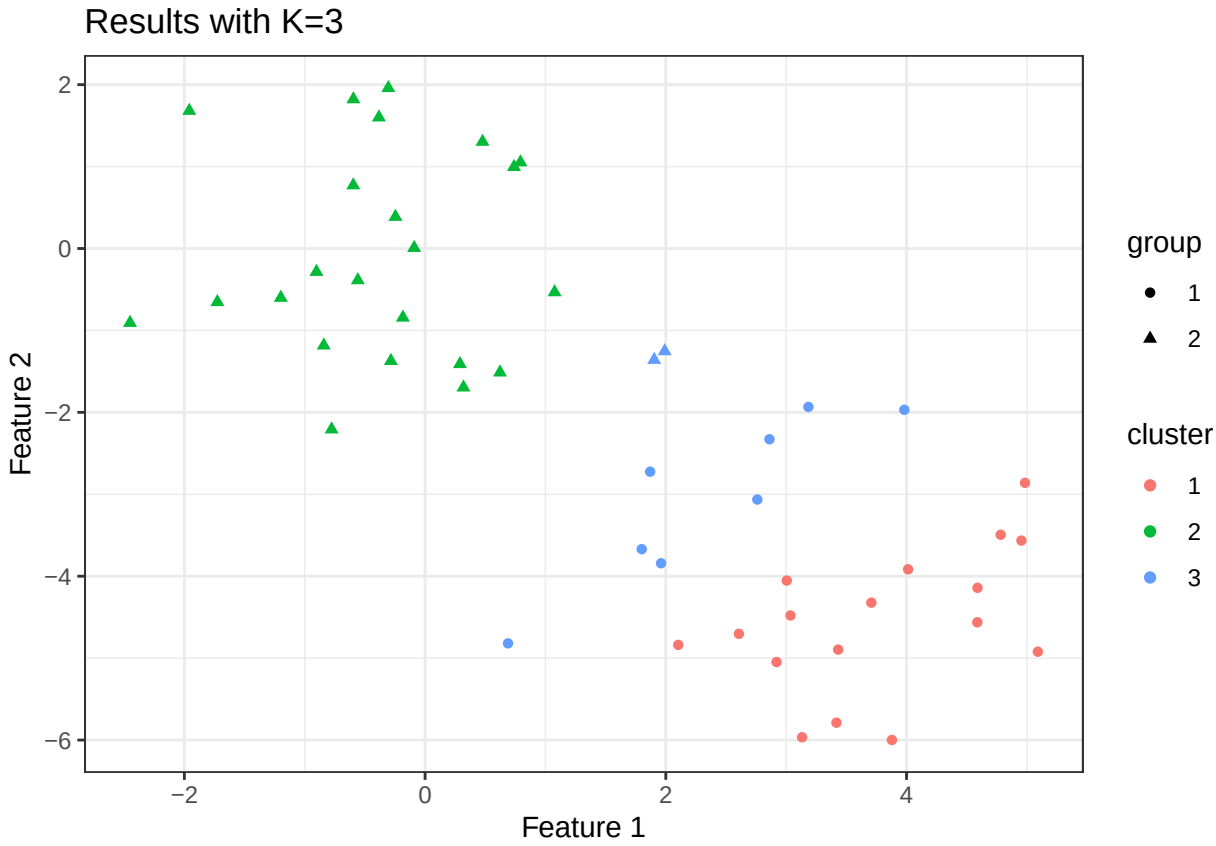
```
# create new data.frame with "group" and "cluster"
y = data.frame(x); colnames(y)=c("X1","X2")
y$group = factor(rep(c(1,2),each=25))
y$cluster=factor(km.out$cluster)
library(ggplot2)
p = ggplot(y,aes(X1,X2)) + geom_point(aes(shape=group,color=cluster)) + xlab("Feature 1")+ylab("Feature
p
```




```
set.seed(4)
km.out3 =kmeans(x,3,nstart=20)
# show the 3 centroids
km.out3$centers
```

```
##          [,1]        [,2]
## 1  3.7789567 -4.56200798
## 2 -0.3820397 -0.08740753
## 3  2.3001545 -2.69622023
```

```
# create new data.frame with "group" and "cluster"
y = data.frame(x); colnames(y)=c("X1","X2")
y$group = factor(rep(c(1,2),each=25))
y$cluster=factor(km.out3$cluster)
p = ggplot(y,aes(X1,X2))+geom_point(aes(shape=group,color=cluster))+xlab("Feature 1")+ylab("Feature 2")
p
```




when an extra cluster is added, at least one of the clusters need to be split into the added cluster. The total within cluster sum of squares cannot be smaller if the number of clusters is incorrect.

As you change the number of clusters throughout the clustering process you will see the within-cluster sum of squares change. You know you likely have the right number of clusters when improvement based on that metric stops.

```
set.seed(3)
km.out = kmeans(x, 3, nstart = 1)
km.out=kmeans(x,3,nstart=1)
km.out$tot.withinss
```

```
## [1] 99.88436
```

```
km.out=kmeans(x,3,nstart=20)
km.out$tot.withinss
```

```
## [1] 97.97927
```

see how the within sum of squares changes with nstart changing.

**Monte Carlo** used for generating data. Started as a mathematics thing for computing complex integrals. Monte-carlo methods refer to situations where you generate a bunch of samples from a distribution then transform the samples and compute something.

**Gap statistic** assumes a null distribution (the null distribution is the distribution of data given the null hypothesis is true). if null distribution is true -> compute the statistic if the null distribution is not true -> ??? Monte carlo is used to compute the statistic. After, compare the stat under the null hypothesis then compare it to the observed statistic. large difference -> reject $H_0$ small difference -> retain $H_0$

clusGap is the function it returns: - tab: a matrix with k.max rows and 4 columns logW, E.logW, gap, and SE.sim Once clusGap has been executed we return the true number $K_0$ of clusters by executing the function maxSE. The best method is "Tibs2001SEmax" but "firstSEmax" is the default. gap statistic assumes a uniform distribution

```
library(cluster)
set.seed(3.14)
gap = clusGap(iris[,1:2], kmeans, K.max = 10, B = 250, nstart = 20, iter.max = 20)
gap$tab
```

```
## NULL
```

```
k = maxSE(gap$Tab[, "gap"], gap$Tab[, "SE.sim"], method="Tibs2001SEmax")
print(k)
```

```
## [1] 3
```