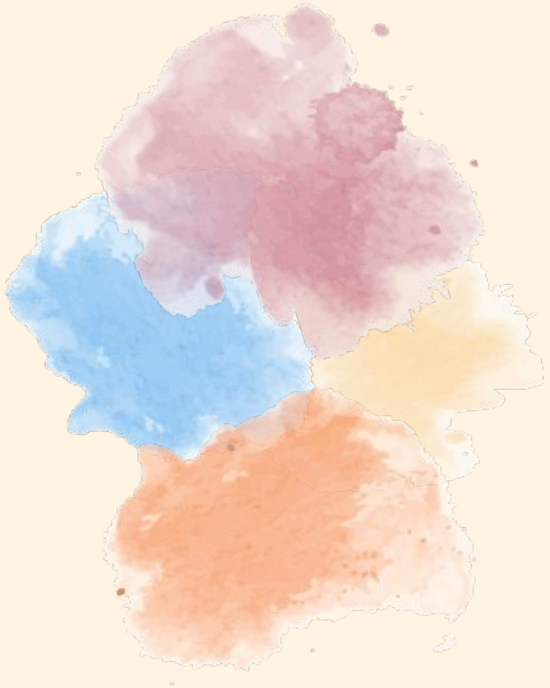
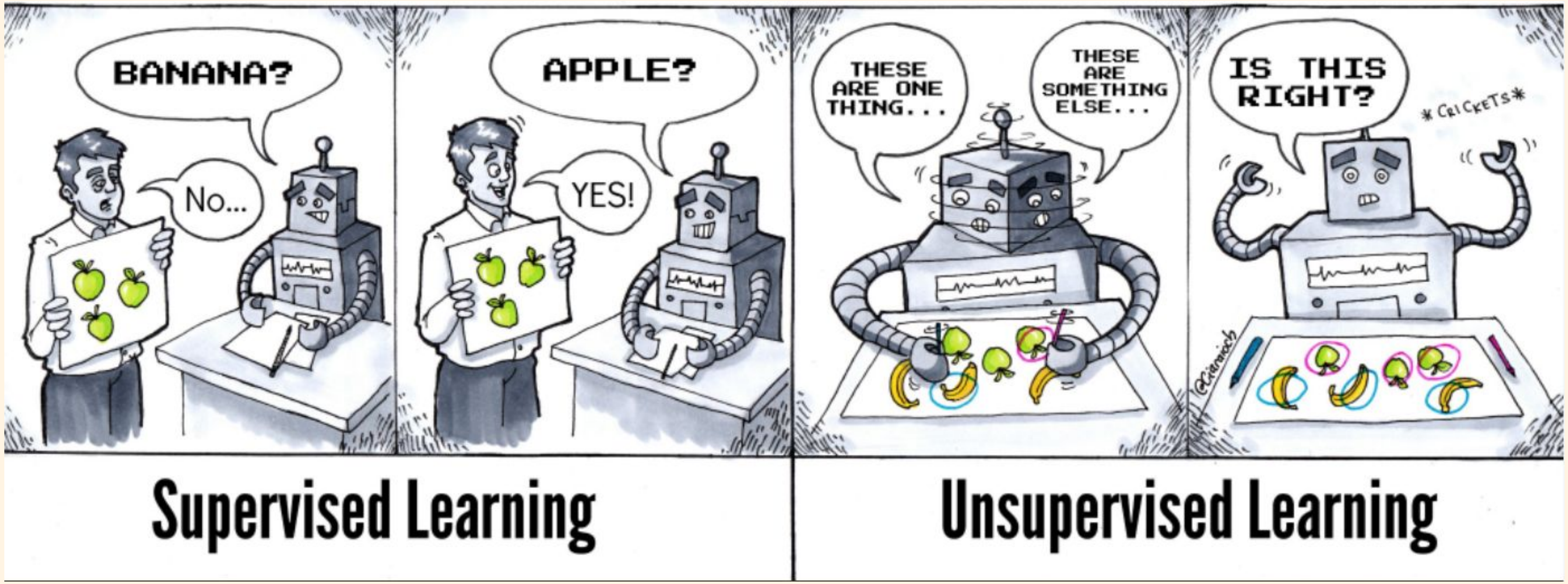
Unsupervised Learning

Part 1: PCA,CA,K-means



Mira
09/04/2022

Unsupervised Learning ?



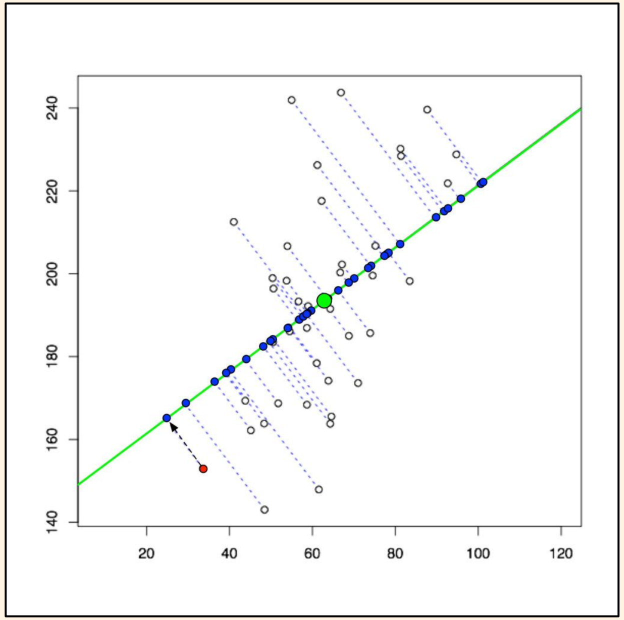
Unsupervised Learning ?

- Algorithms work on their own to discover the structure of unlabeled data, the hidden patterns or data groupings (similarities and differences in information) without the need for human intervention.
- Those are used for three main tasks: **Clustering, association and dimensionality reduction.**

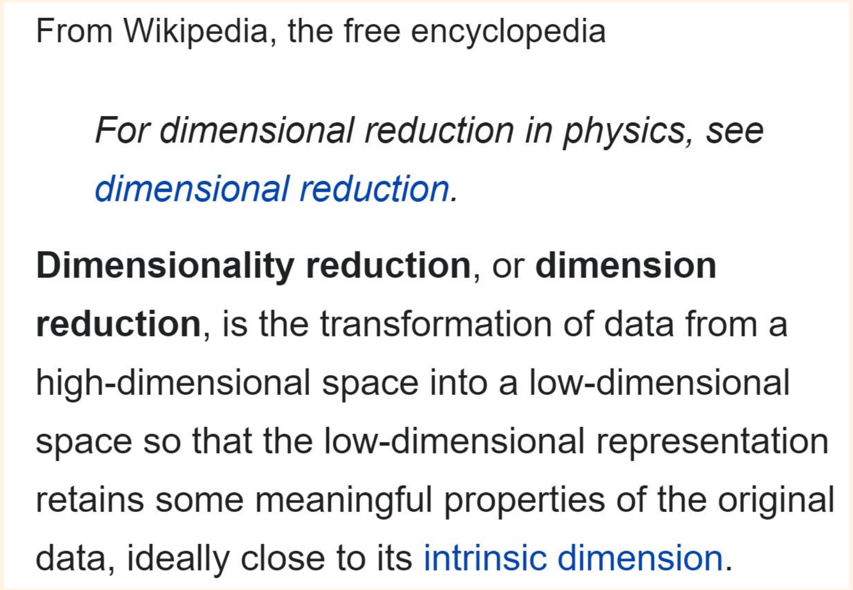
Table of contents

01. Dimensionality reduction

PCA,Correspondence Analysis



Definition



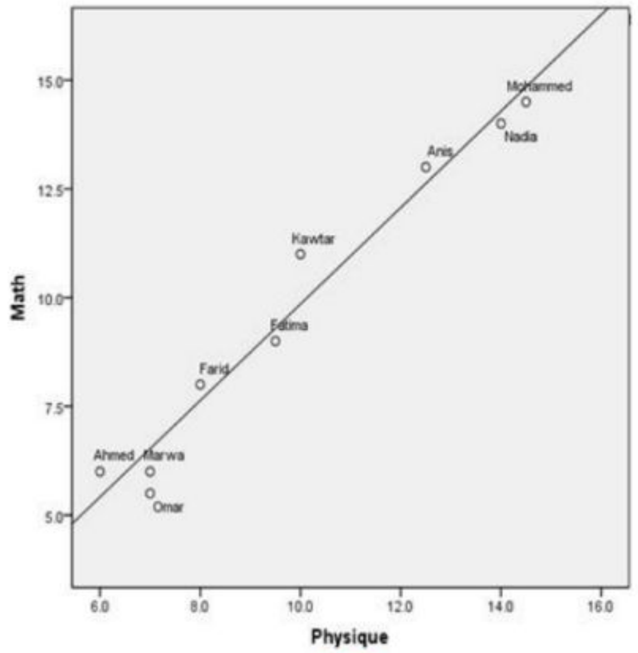
From Wikipedia, the free encyclopedia

For dimensional reduction in physics, see dimensional reduction.

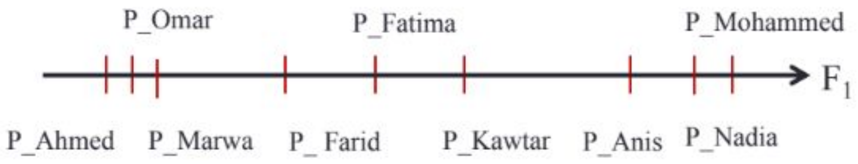
Dimensionality reduction, or **dimension reduction**, is the transformation of data from a high-dimensional space into a low-dimensional space so that the low-dimensional representation retains some meaningful properties of the original data, ideally close to its intrinsic dimension.

Intrinsic dimension is the number of variables needed for minimal representation of the data.

- Reducing the dimensions/features of a dataset with ensuring most of the key information is maintained.
- Used to impact the performance of the model, minimise its computational complexity and avoid falling into overfitting.
- Its Methods: **PCA, Correspondence Analysis, Singular value decomposition, Autoencoders.**


Math
Physique
15.0
12.5
10.0
7.5
5.0
6.0
8.0
10.0
12.0
14.0
16.0
Mohammed
Nadia
Anis
Kawtar
Fatima
Farid
Ahmed
Marwa
Omar

2 D


P_Omar
P_Fatima
P_Mohammed
F_1
P_Ahmed
P_Marwa
P_ Farid
P_Kawtar
P_Anis
P_Nadia

1 D

Principal Components Analysis (PCA)

- This method create a new data representation(a set of "principal components")using a linear transformation.
- **Only** Numerical features
- PCA ensure that the new dimensions maximize the variance in the data(most useful and contain the most information).

Principal component !?!

Notes

- Scaling the vector in blue results into the pink one.

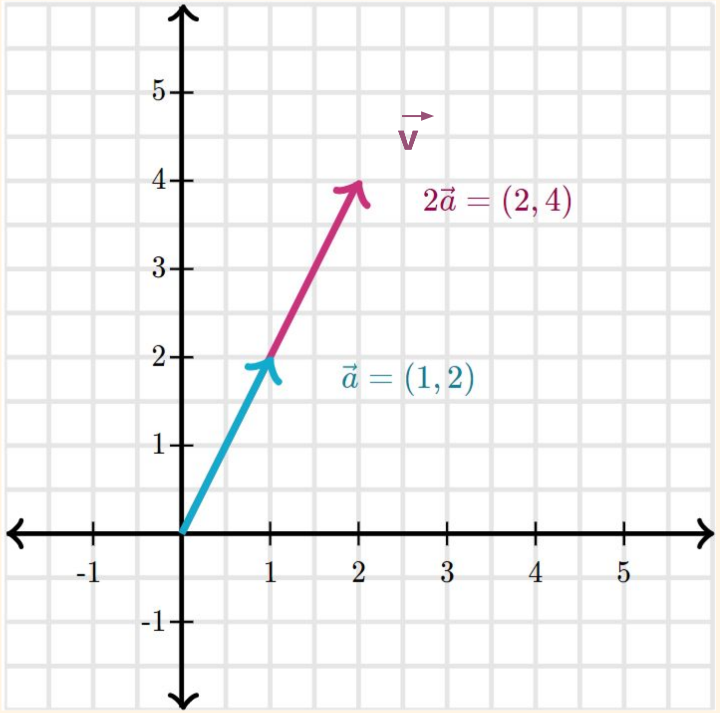
$\vec{v} = 2.\vec{a}$

- **Eigenvectors** :Vectors when we apply a linear transformation to them they become a scaled version of themselves.

Matrix . $\vec{v} = \vec{v} . \lambda$

(λ is a scalar)

- **EigenValues** :Lambda λ scalar/ coefficient that gives the eigenvector its magnitude.



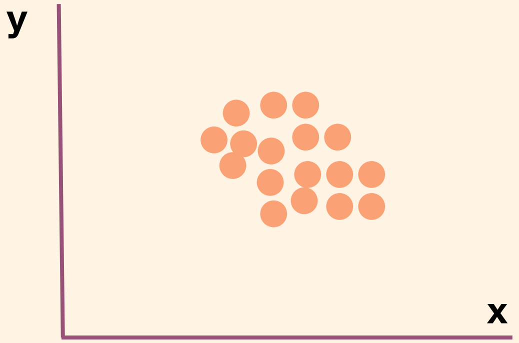
Notes

- Covariance : how each variable is associated with one another .

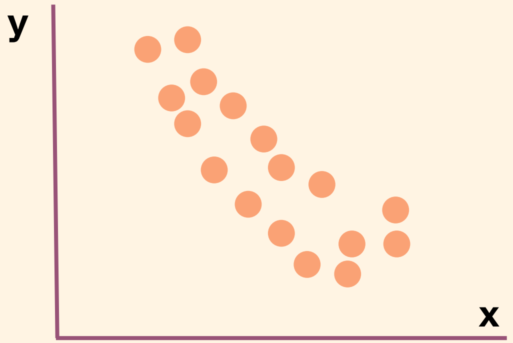
$$\mathrm{Cov}(x,y) = \frac{\Sigma\,(x_i - \bar{x}) * (y_i - \bar{y})}{N}$$

$$\mathrm{cov}(X,X) = \mathrm{var}(X)$$
$$\mathrm{cov}(X,Y) = \mathrm{cov}(Y,X)$$

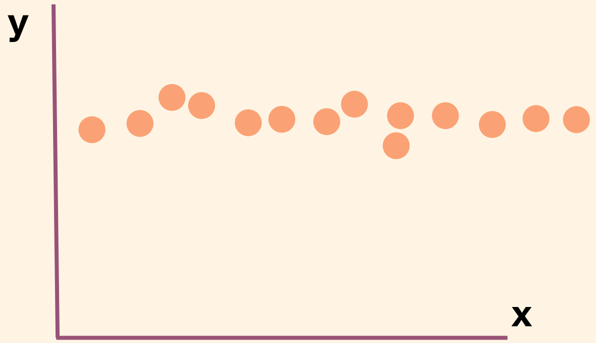
- Variance: How the data is spread around its mean.



Low variance



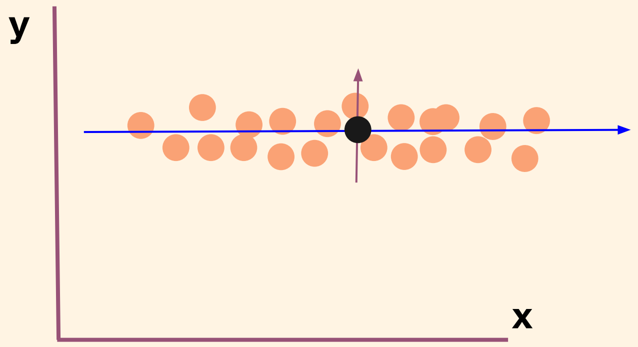
High variance

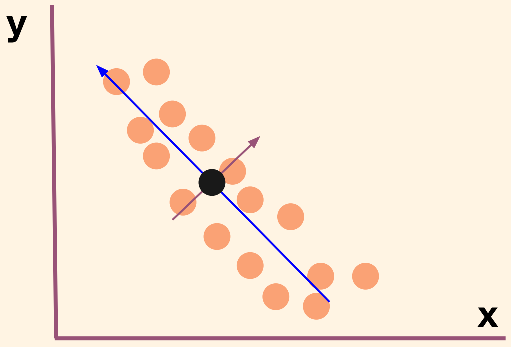


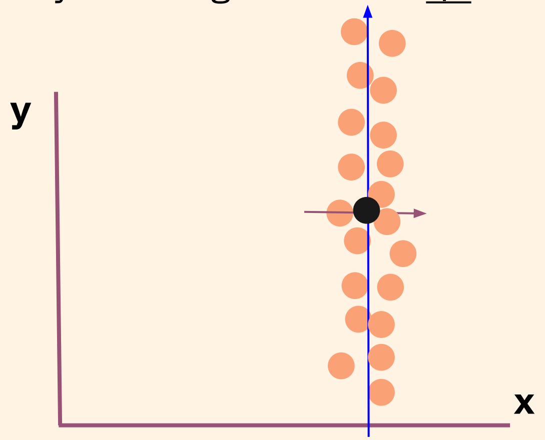
Maximum Variance in x**(axis)**

Principal component ?

- Principal components are new axis/features that represent the maximum variance of the data(best fit lines)that go through the **center** of the data.
- The numbers of principal components that could be found are equal to the number of dimensions.
- The principal components are **uncorrelated**,mutually orthogonal PC1 $\perp$ PC2 and var(PC1)>>var(PC2).







How PCA algorithm works ?

1. Centring and reducing the dataset
2. Calculating covariance matrix Σ
3. EigenDecomposition the cov matrix
4. New basis (eigenvectors) of data presentation
5. Projecting all the data in the new axis

1. Centring and reducing the dataset

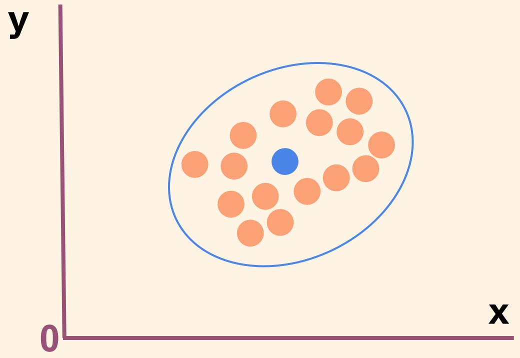
- Centring

Dataset_Matrix:

Rows = observations
Columns = features

	x	y
1	35	16
2	52	23
3	48	23
4	23	14
5	10	32
..	..	..

Mean	$\bar{x}$	$\bar{y}$

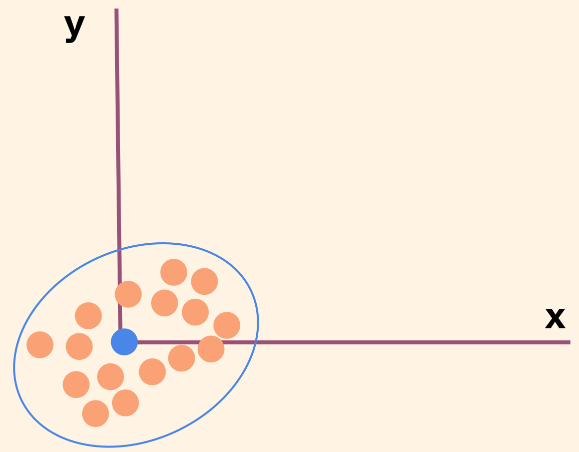


1. Centring and reducing the dataset

- Centring

	x	y
1	$35-\overline{x}$	$16-\overline{y}$
2	$52-\overline{x}$	$23-\overline{y}$
3	$48-\overline{x}$	$23-\overline{y}$
4	$23-\overline{x}$	$14-\overline{y}$
5	$10-\overline{x}$	$32-\overline{y}$
..	..	..

Mean	0	0



1. Centring and reducing the dataset

- Reducing /Scaling

- When PCA tries to get the features with maximum variance and the variance is high for high magnitude features.

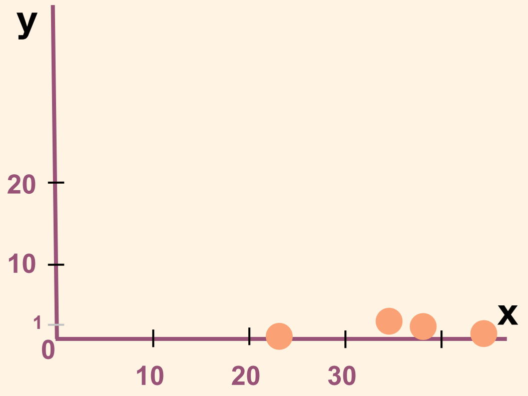
- To avoid biased results of the pca(variance calculation)
the features are scaled to the same range using its standard deviation.

	x	y
1	35	1
2	22	0.23
3	45	0.35
4	38	0.5

Variance	69	0.085

Standard Deviation	√69.5 = 8.33	√0.085 = 0.29

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}.$$

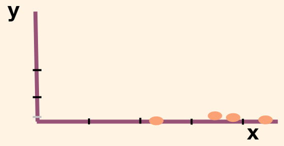


1. Centring and reducing the dataset

- Reducing /Scaling

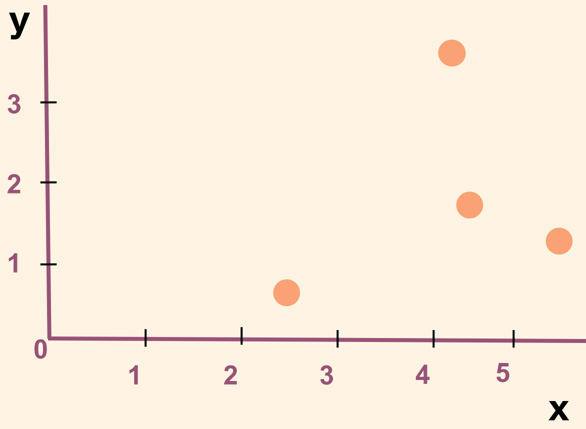
	x	y
1	35	1
2	22	0.23
3	45	0.35
4	38	0.5

Standard Deviation	√69.5 = 8.33	√0.085 = 0.29



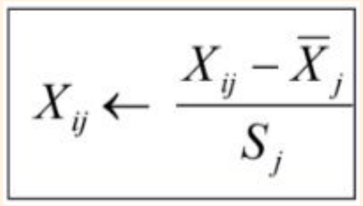
	x	y
1	4.2	3.44
2	2.64	0.79
3	5.4	1.2
4	4.56	1.7

Variance	1.0008	1.0196



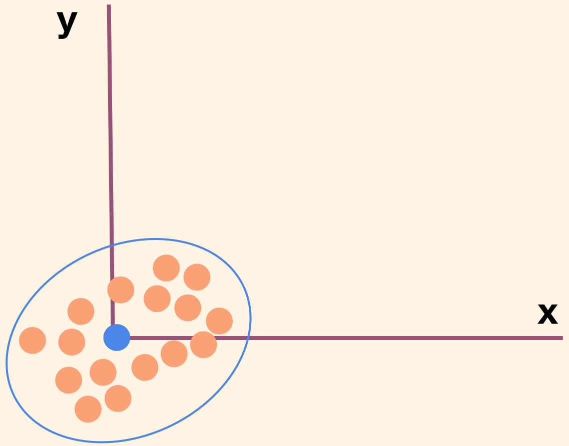
1. Centring and reducing the dataset

- Centering and Reducing



$$X_{ij} \leftarrow \frac{X_{ij} - \bar{X}_j}{S_j}$$

	x	y
i	$\frac{x_i - \bar{x}}{\sigma_x}$	$\frac{y_i - \bar{y}}{\sigma_y}$
..	..	..



2. Calculating covariance matrix Σ

$$\mathrm{Var}_X(j) = \frac{1}{n}\sum_i (\bar{x}_j - X_{ij})^2$$

$$\mathrm{Cov}_X(j,k) = \frac{1}{n}\sum_i (\bar{x}_j - X_{ij})(\bar{x}_k - X_{ik})$$

After centering the data $\longrightarrow$

$$\mathrm{Var}_X(j) = \frac{1}{n}\sum_i X_{ij}X_{ij}$$

$$\mathrm{Cov}_X(j,k) = \frac{1}{n}\sum_i X_{ij}X_{ik}\,.$$

- The covariance matrix defines both the spread (variance), and the orientation (covariance) of our data.

$$\begin{array}{c|cc} & x & y \\ \hline x & var(x) & cov(x,y) \\ y & cov(x,y) & var(y) \end{array}$$

2D

$$\begin{array}{c|ccc} & x & y & z \\ \hline x & var(x) & cov(x,y) & cov(x,z) \\ y & cov(x,y) & var(y) & cov(y,z) \\ z & cov(x,z) & cov(y,z) & var(z) \end{array}$$

3D

3. Decomposing the Σ matrix

- The principal components are defined by the eigenvectors and the eigenvalues of the covariance matrix.

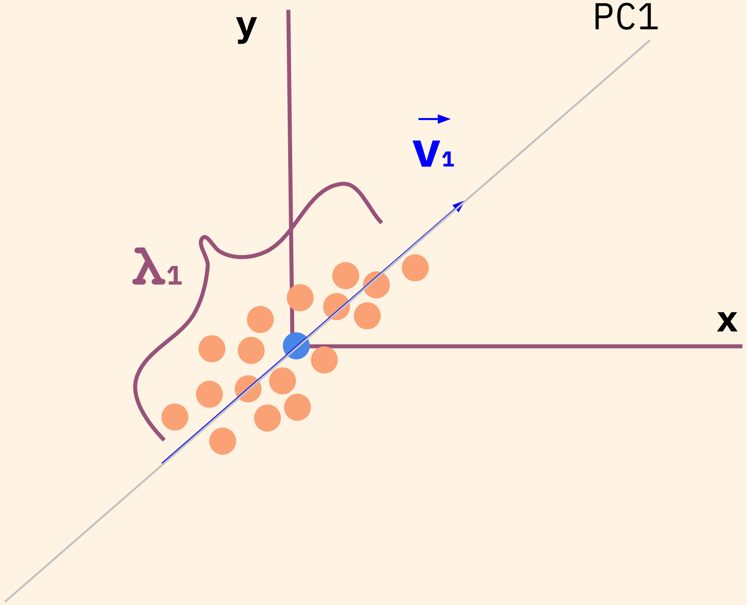
- Eigenvalues represent the variance of the data on each eigenvector.

$$\det(\Sigma - I\lambda) = 0$$

=> λ_j (j is the number of dimensions)

$$\Sigma \cdot \vec{v} = \vec{v} \cdot \lambda$$

(λ is a scalar)



3. Decomposing the Σ matrix

- Now sorting the eigenvalues in a decreasing order so the first eigenvector hold the max information/variance/eigenvalue $\lambda_1 >> \lambda_2$

$$\boxed{\lambda_j >> \lambda_{j-1}}$$

- Then calculating the new eigenvectors by solving this linear system :

$$\Sigma \cdot \vec{v} = v \cdot \lambda$$

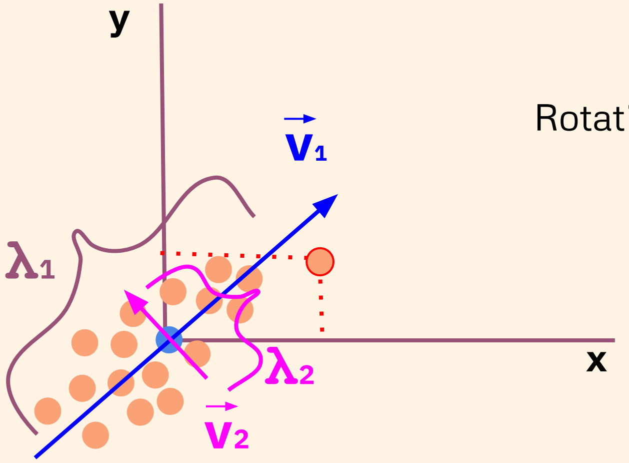
$\Rightarrow \vec{v}_j$ **(j is the number of dimensions)**

4. New basis of data presentation

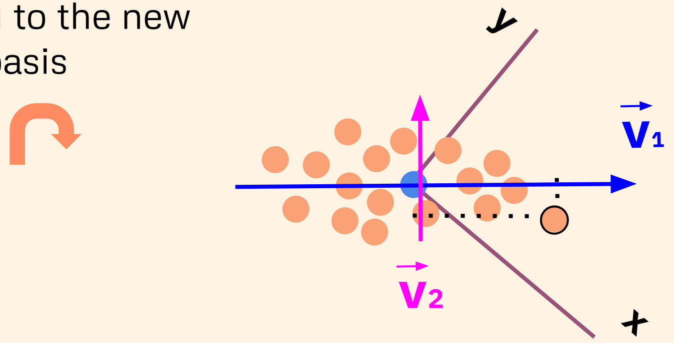
- The new basis are our new eigenvectors

$\vec{V} = (\vec{v}_1, \vec{v}_2)$

$\vec{v}_j$ (j is the number of dimensions)



Rotating to the new basis



5. Projecting in the new basis

- The projection(dot product) of the original observations to the new basis will give as the principal components.

$\vec{PC}_j = \vec{v}_j \cdot \text{Dataset\_Matrix}$
(j is the number of dimensions)

- When solving this equation $\mathbf{v}_j$(a,b,c,…)are now called loadings of the principal components.

V_j
a
b
c
…

- The PCs are represented as follow: **PCj** $= a x_i + b y_i + c z_i + ..$

5. Projecting in the new basis

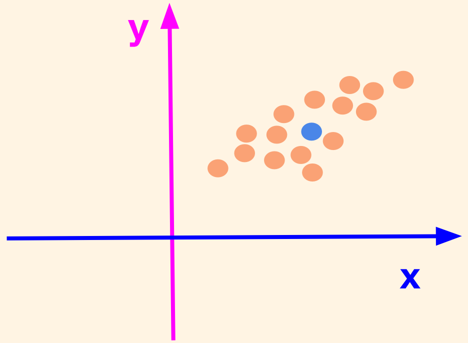
V_1
a
b

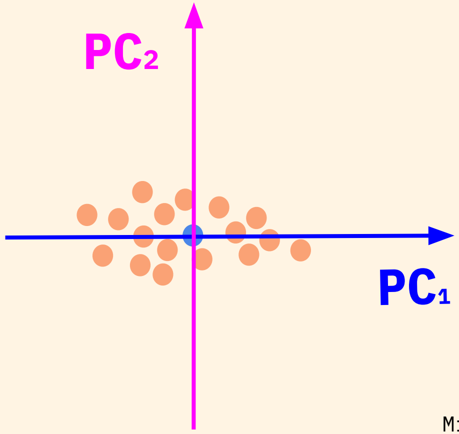
V_2
a'
b'

	x	y
i	x_i	y_i
..	..	..

New representation

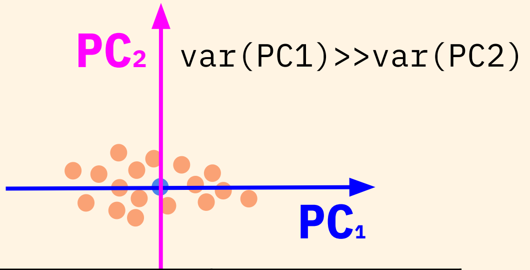
	PC_1	PC_2
i	ax_i+by_i	$a'x_i+b'y_i$
..	..	..





Can’t see the dimensionality reduction in PCA

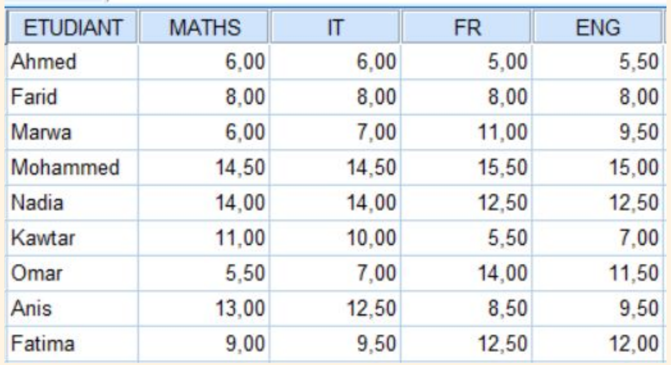
Reducing the PCs ?



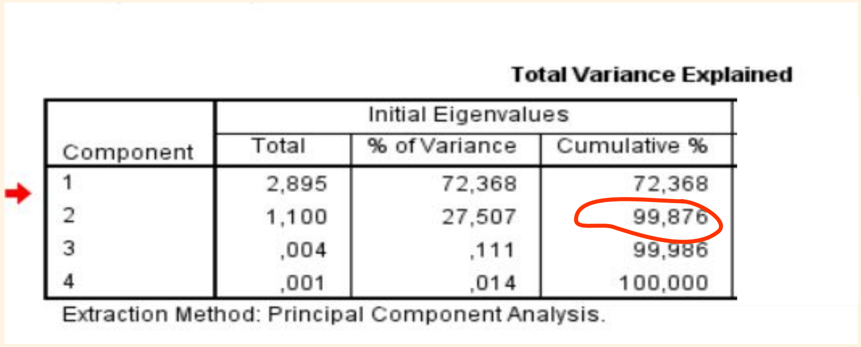
- We need to select k PCs from the n ones(number of dimensions)with **k<n**.
- The selection is based on the maximum variance explained by each PC.
- The variance of each PC is the eigenvalue related to it (λ).
- We don't need to have 100% of the information explained by a subspace !

	PC_1	PC_2
i	ax_i+by_i	$a'x_i+b'y_i$
Variance	λ_1	λ_2
% of var	$a=\frac{\lambda_1}{Sum(\lambda_i)}*100$	$b=\frac{\lambda_2}{Sum(\lambda_i)}*100$
Cumulative	**a**	**a+b**

Example



ETUDIANT	MATHS	IT	FR	ENG
Ahmed	6,00	6,00	5,00	5,50
Farid	8,00	8,00	8,00	8,00
Marwa	6,00	7,00	11,00	9,50
Mohammed	14,50	14,50	15,50	15,00
Nadia	14,00	14,00	12,50	12,50
Kawtar	11,00	10,00	5,50	7,00
Omar	5,50	7,00	14,00	11,50
Anis	13,00	12,50	8,50	9,50
Fatima	9,00	9,50	12,50	12,00



Total Variance Explained

Component	Initial Eigenvalues		
	Total	% of Variance	Cumulative %
1	2,895	72,368	72,368
2	1,100	27,507	99,876
3	,004	,111	99,986
4	,001	,014	100,000

Extraction Method: Principal Component Analysis.

- Sine the subspace(PC1,PC2) explains more than 99% of variance,those two dimensions will be sufficient to represent the data.

Correspondence Analysis

- **CA is not a PCA for categorical data !**
- It’s visualisation technique, that helps to explore and analyze the relation **between** two categorical features.
- It’s also a dimension reduction tool applied to contingency tables.

Contingency table ?

Contingency table ?

	Gender	Field of studies
1	F	Physics
2	F	Physics
3	M	Biology
4	F	Art
5	M	Biology
6	M	Biology
7	F	Art
8	M	Art
9	M	Physics
10	F	Biology

Contingency/cross table

Field of studies \ Gender	F	M
Physics	2	1
Biology	1	3
Art	2	1

How CA works ?

1. Row and Column Profile tables
2. Interdependence Test
3. Dimensionality reduction
4. Row/Column Plotting

1. Row and Column profile tables

- In order to have row-profiles and column profile tables we calculate conditional frequencies.

$$f_{j|i} = P(V_2 = j | V_1 = i) = \frac{P(V_2 = j \cap V_1 = i)}{P(V_1 = i)}$$

$$f_{j|i} = \frac{\frac{n_{ij}}{n}}{\frac{n_{i.}}{n}} = \frac{n_{ij}}{n_{i.}}$$

Hair Color / Eye Color	blond	red	brunette	**Total ($n_{i.}$)**
Blue	10	10	10	30
Brown	7	6	7	20
Green	13	4	33	50
Total($n_{.j}$)	30	20	50	**n** =100

$f_{i.} = n_{i.} / n$
3/10
1/5
1/2

Relative Frequency $f_{.j} = n_{.j} / n$	3/10	1/5	1/2

Row-profile table

Hair Color / Eye Color	blond	red	brunette
Blue	1/3	1/3	1/3
Brown	7/20	6/20	7/20
Green	13/50	4/50	33/50

1. Row and Column profile tables

Hair Color / Eye Color	blond	red	brunette	**Total**
Blue	10	10	10	30
Brown	7	6	7	20
Green	13	4	33	50
Total	30	20	50	100

Row-profile table

Hair Color / Eye Color	blond	red	brunette
Blue	1/3	1/3	1/3
Brown	7/20	6/20	7/20
Green	13/50	4/50	33/50

Column-profile table

Hair Color / Eye Color	blond	red	brunette
Blue	1/3	1/2	1/5
Brown	7/30	6/20	7/50
Green	13/30	4/20	33/50

2. Independence Test

- Using Chi square:

$$\chi^2 = \sum_{i=1}^{I}\sum_{j=1}^{J}\frac{\left(n_{ij} - \frac{n_{i.}n_{.j}}{n}\right)^2}{\frac{n_{i.}n_{.j}}{n}}$$

- Hypothesis:

  H_0:The rows and the columns are independent.

  H_a:There is a link between the rows and columns are.

- Degree of freedom :

Df = (nb of rows-1)*(nb of col -1)

If there is no Link no need to continue the CA !

3. Dimensionality reduction

Hair Color / Eye Color	blond	red	brunette	**Total ($n_{i.}$)**
Blue	10	10	10	30
Brown	7	6	7	20
Green	13	4	33	50
Total($n_{.j}$)	30	20	50	**n** =100

$f_{i.} = n_{i.} / n$
3/10
1/5
1/2

Relative Frequency $f_{.j} = n_{.j} / n$	3/10	1/5	1/2

D_I the matrix of row-weights

$$\begin{Bmatrix} 3/10 & 0 & 0 \\ 0 & 1/5 & 0 \\ 0 & 0 & 1/2 \end{Bmatrix}$$

$$\begin{Bmatrix} 3/10 & 0 & 0 \\ 0 & 1/5 & 0 \\ 0 & 0 & 1/2 \end{Bmatrix}$$

D_J the matrix of column-weights

3. Dimensionality reduction

- H is our matrix for the eigenanalysis:

$$\mathbf{H} = \mathbf{D}_J^{1/2}\mathbf{Z}^T\mathbf{D}_I\mathbf{Z}\mathbf{D}_J^{1/2}$$

- D_I the matrix of row-weights
- D_J the matrix of columns-weights
- Z table of contingency

- To calculate Eigenvalues

```
Trace(H)= Sum(λn)
=>λn  (λ # 1)
```

$$\left\{\begin{matrix} a & b & c \\ d & e & f \\ g & g & h \end{matrix}\right\}$$

Trace =a+b+c

- Depending on the resulted Eigenvalues we'll have eigenvectors.

3. Dimensionality reduction

- Depending on the variance explained/inertia by each eigenvectors (factors/new basis) we'll reduce our dimensions/factors (same as reducing PCs in PCA).
- For this cross table (Hair color/Eyes color)the results will be as follow :

Calculations are not correct here

~~$\lambda_0 = 1$~~ $(\lambda \# 1)$

$\lambda_1 = 0.8937$ =>v1

$\lambda_2 = 0.095$ =>v2

$\lambda_3 = 0.011$ =>v3

	F(v1)	F(v2)	F(v3)
var= λ	0.8937	0.095	0.011
λ*100/SUM(λi)	89.39	9.50	1.1
Cumulative	89.39	98.89	99.99

=>The space will be reduced to 2 Dimensions (F1,F2)

4. Row/Column Plotting

```
v1,v2
v = (a,b,..)
```

- Projecting the columns on the eigenvectors will result to columns-factors.

$$D = v.\sqrt{\lambda}$$

- Projecting the rows on the eigenvectors will result to row-factors.

$$C = (\text{Row-profileMatrix}).v$$

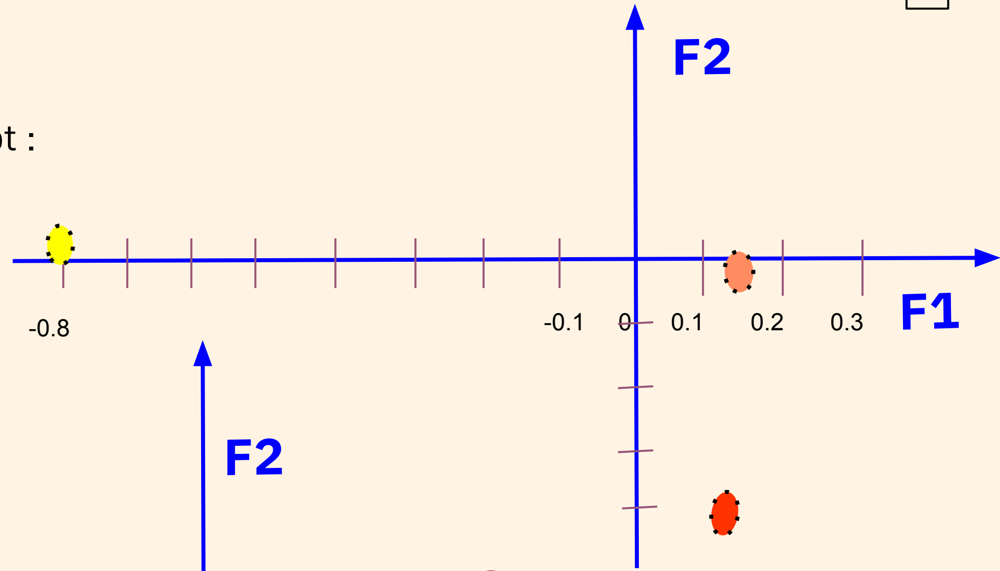
Factors / Hair Color	F1(D1)	F2(D2)
blond	-0.835	0.0695
red	0.1482	-0.032
brunette	0.1295	-0.3196

Factors / Eye Color	F1(C1)	F2(C2)
Blue	-0.5474	0.0829
Brown	0.492	0.088
Green	-0.1617	-0.339

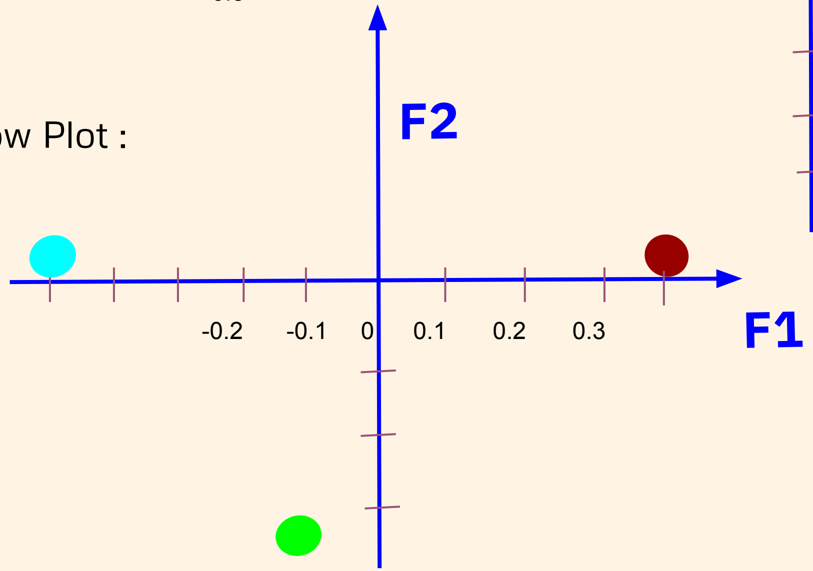
Factors / Hair Color	**F1(D1)**	**F1(D1)**
blond	-0.835	0.0695
red	0.1295	-0.3196
brunette	0.1482	-0.032

Factors / Eye Color	**F1(C1)**	**F2(C2)**
Blue	-0.5474	0.0829
Brown	0.492	0.088
Green	-0.1617	-0.339

- Column Plot :



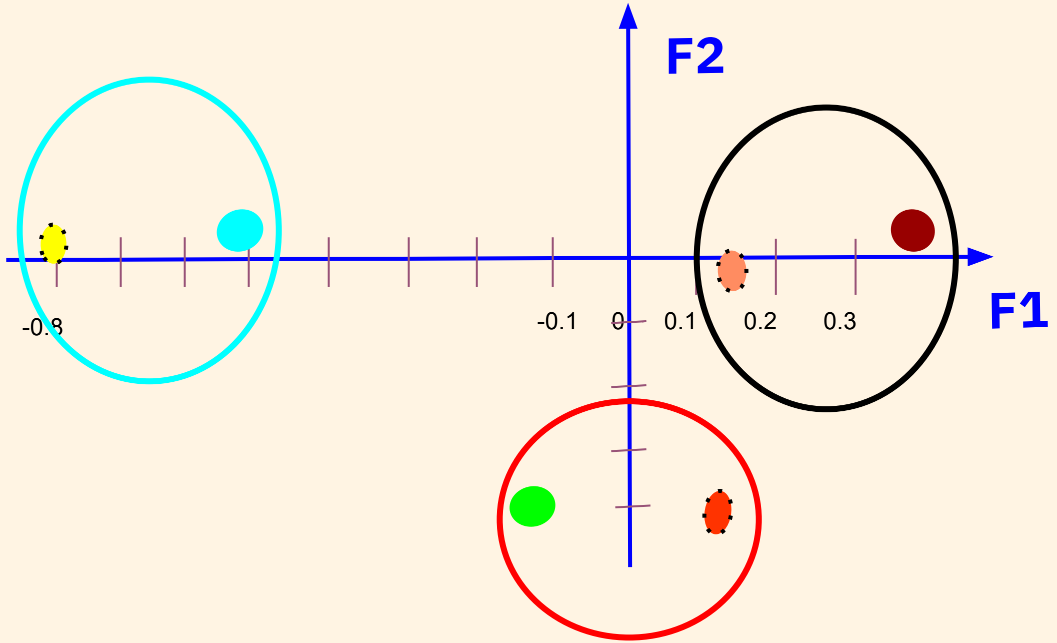
- Row Plot :



Factors / Hair Color	F1(D1)	F1(D1)
blond	-0.835	0.0695
red	0.1295	-0.3196
brunette	0.1482	-0.032

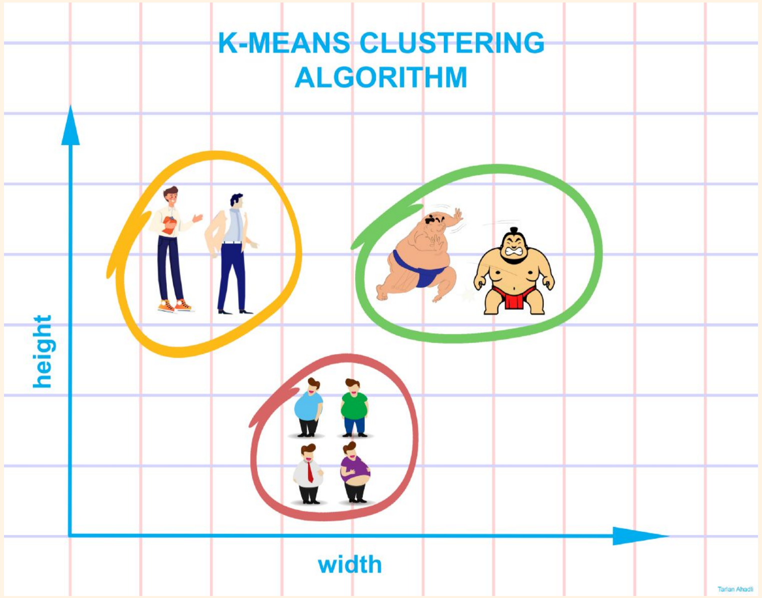
Factors / Eye Color	F1(C1)	F2(C2)
Blue	-0.5474	0.0829
Brown	0.492	0.088
Green	-0.1617	-0.339

- Row-Column Plot :



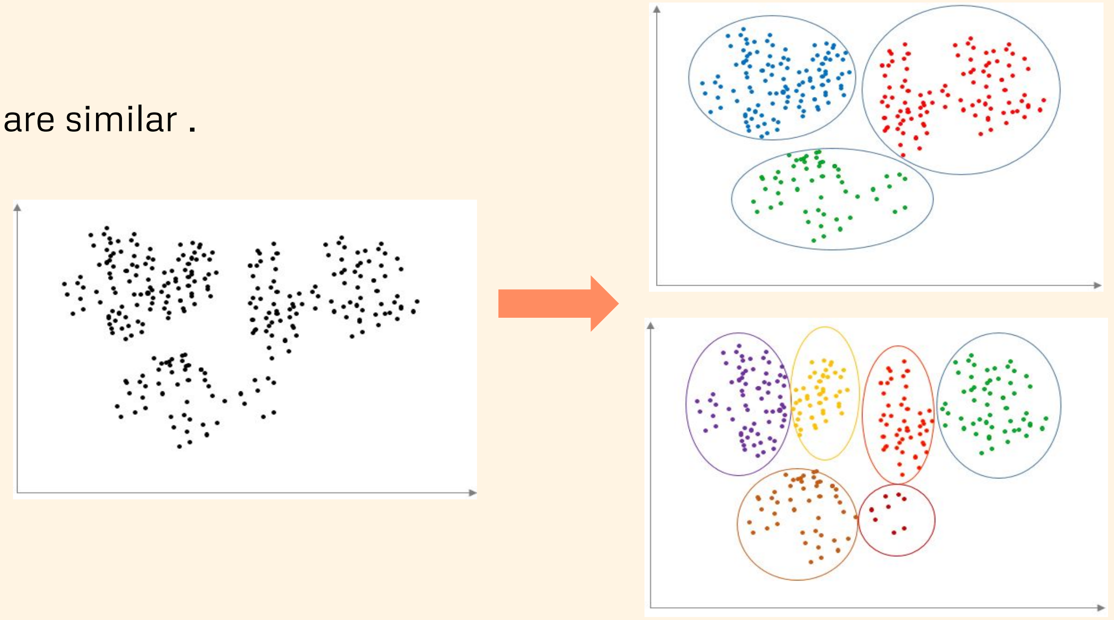
02. Clustering

K-means



K-means

- Algorithm aims to partition n observations into k clusters using an iterative method.
- Cluster is a group of observations that are similar .
- Each cluster is defined by its centre.
- Each observation belongs to cluster whose center is the closest one.

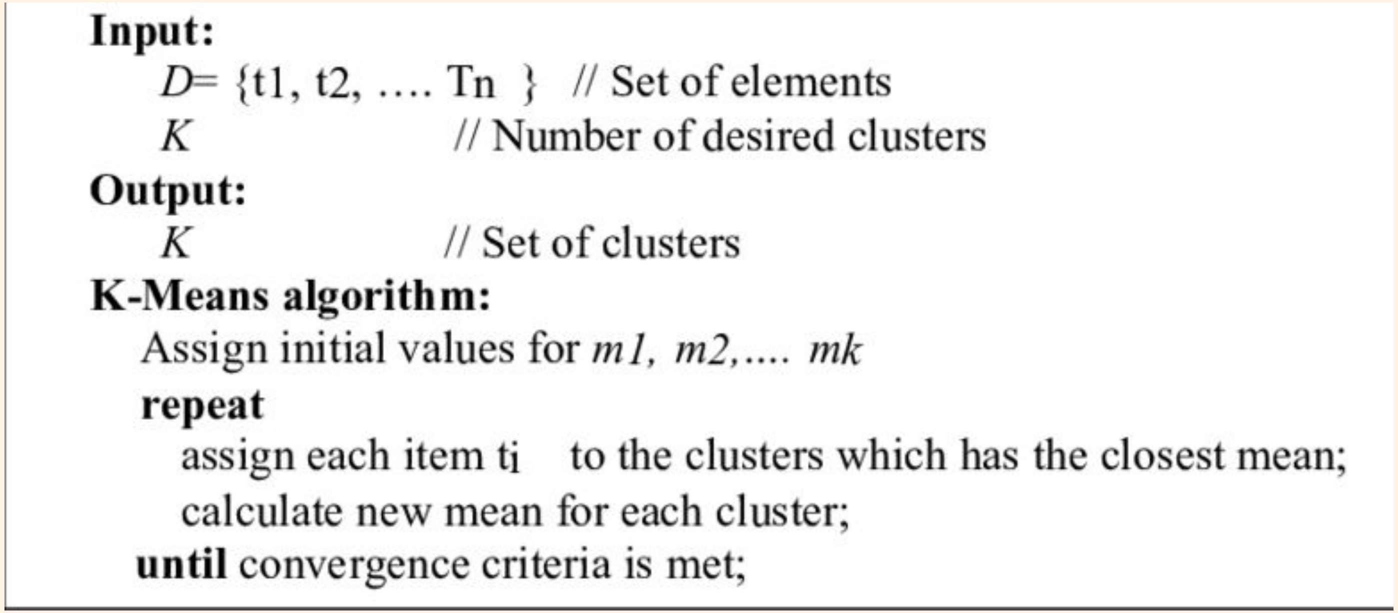


K-means

- Distance metrics plays a very important role in the k-means clustering process, used to find similar data objects.
- The less distance between two objects means they belong to the same cluster.
- Different distance metrics : **Euclidiean Distance, Mahatan Distance, Chebychev Distance, Minkowski Distance** ...
- K-means goal is to minimize the sum of squared error **SSE** over the clusters.

SSE is the sum of squared distances between the center and the points of a cluster.

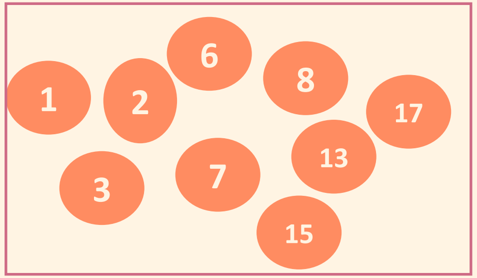
How K-means work ?



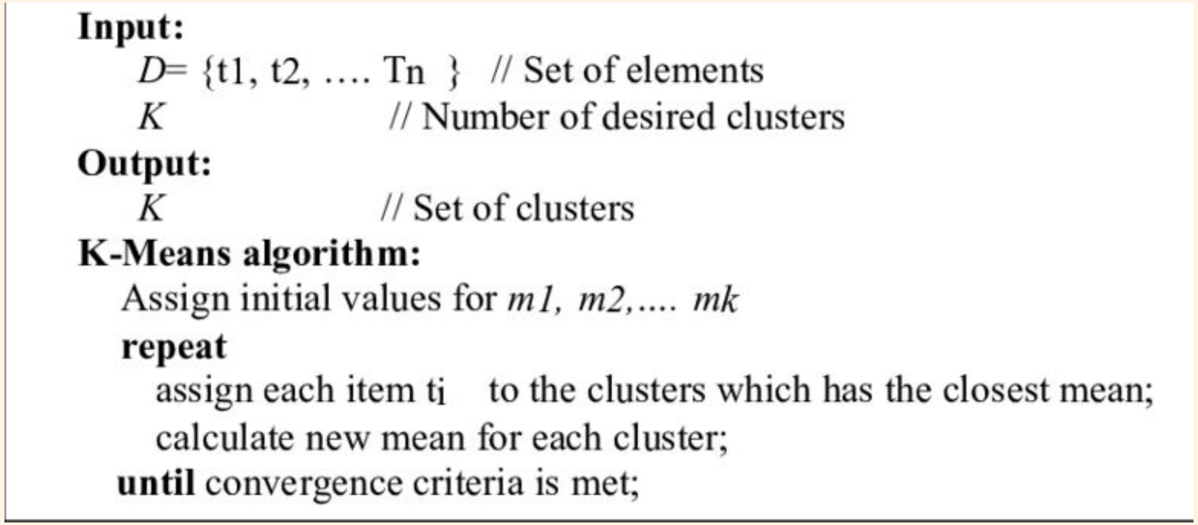
```
Input:
    D= {t1, t2, .... Tn }   // Set of elements
    K                       // Number of desired clusters
Output:
    K                       // Set of clusters
K-Means algorithm:
  Assign initial values for m1, m2,.... mk
  repeat
    assign each item ti   to the clusters which has the closest mean;
    calculate new mean for each cluster;
  until convergence criteria is met;
```

Example

- Dataset A={1,2,3,6,7,8,13,15,17}



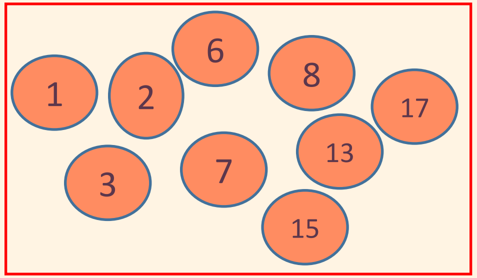
- Let's create 3 clusters :

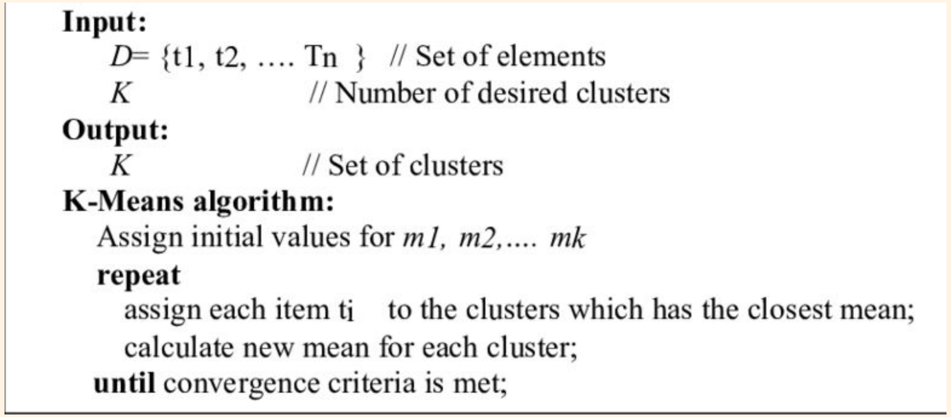


```
Input:
    D= {t1, t2, .... Tn }   // Set of elements
    K                       // Number of desired clusters
Output:
    K                       // Set of clusters
K-Means algorithm:
    Assign initial values for m1, m2,.... mk
    repeat
        assign each item ti   to the clusters which has the closest mean;
        calculate new mean for each cluster;
    until convergence criteria is met;
```

Example

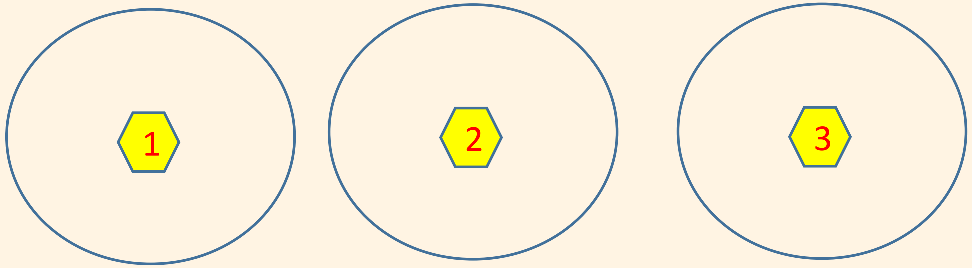
- **Dataset**





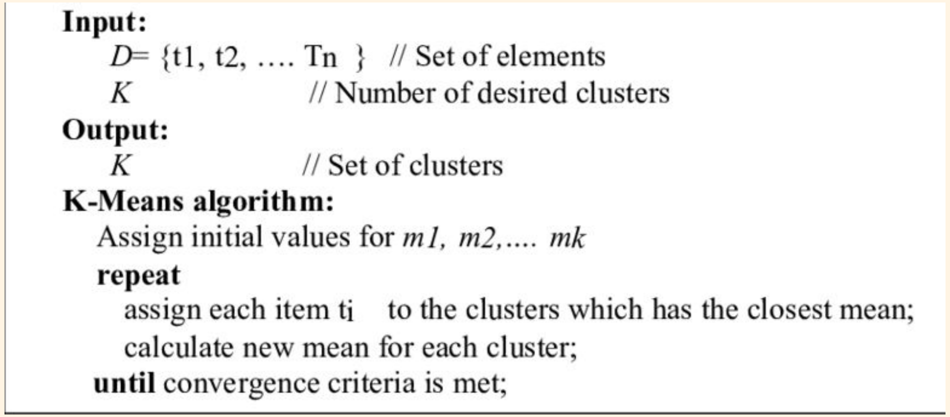
```
Input:
    D= {t1, t2, .... Tn }   // Set of elements
    K                       // Number of desired clusters
Output:
    K                       // Set of clusters
K-Means algorithm:
  Assign initial values for m1, m2,.... mk
  repeat
    assign each item ti   to the clusters which has the closest mean;
    calculate new mean for each cluster;
  until convergence criteria is met;
```

- **We select 3 objects (1, 2 et 3) randomly to create 3 clusters:**



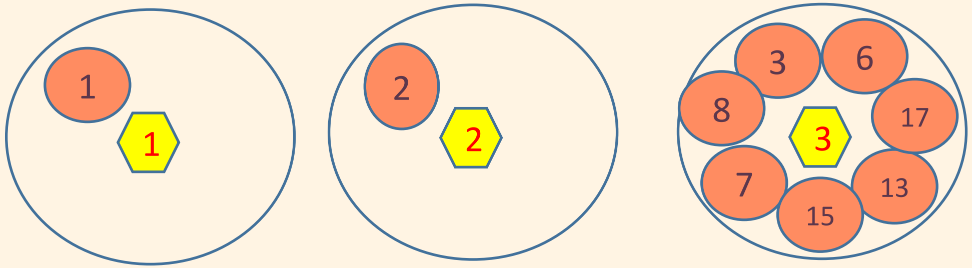
Example

- **Dataset**



```
Input:
    D= {t1, t2, .... Tn }   // Set of elements
    K                       // Number of desired clusters
Output:
    K                       // Set of clusters
K-Means algorithm:
  Assign initial values for m1, m2,.... mk
  repeat
    assign each item ti   to the clusters which has the closest mean;
    calculate new mean for each cluster;
  until convergence criteria is met;
```

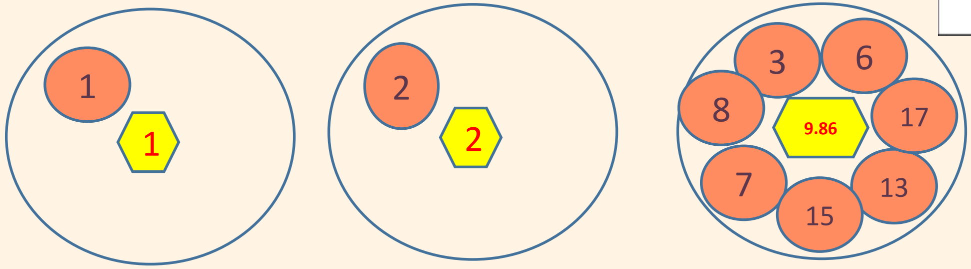
- **Assign each object to its closest cluster**

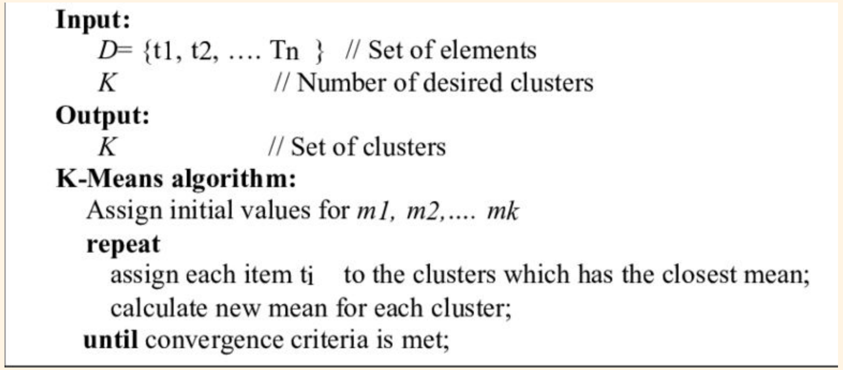


Example

- **Dataset**

- **Calculate the center**

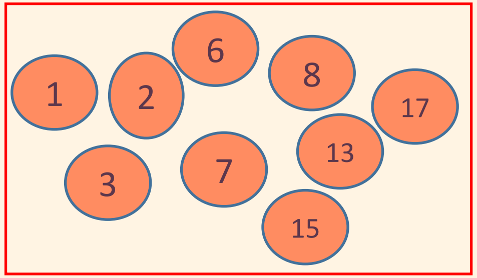




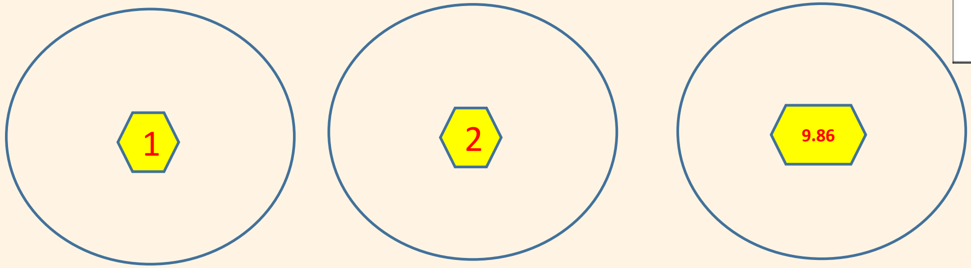
```
Input:
    D= {t1, t2, .... Tn }   // Set of elements
    K                       // Number of desired clusters
Output:
    K                       // Set of clusters
K-Means algorithm:
  Assign initial values for m1, m2,.... mk
  repeat
    assign each item ti   to the clusters which has the closest mean;
    calculate new mean for each cluster;
  until convergence criteria is met;
```

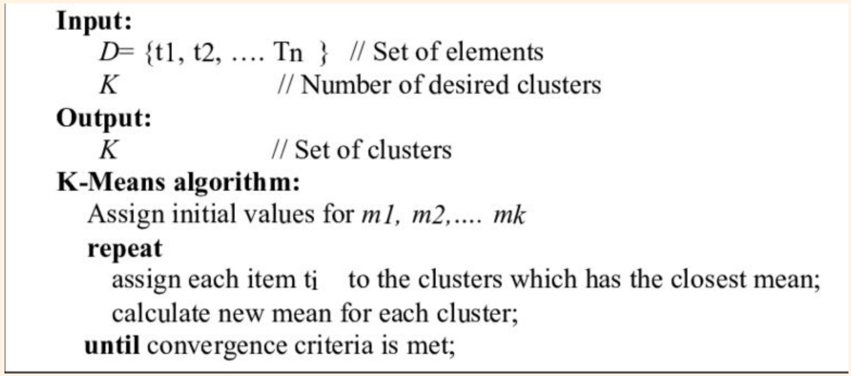
Example

- **Dataset**



- **Reassign each object to its closest cluster**

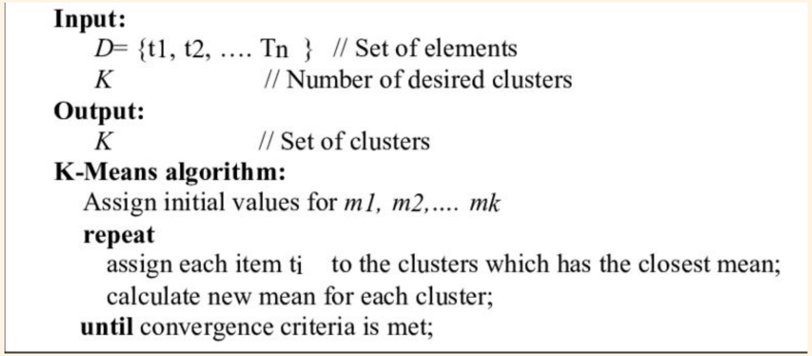




```
Input:
    D= {t1, t2, .... Tn }   // Set of elements
    K                       // Number of desired clusters
Output:
    K                       // Set of clusters
K-Means algorithm:
  Assign initial values for m1, m2,.... mk
  repeat
    assign each item ti  to the clusters which has the closest mean;
    calculate new mean for each cluster;
  until convergence criteria is met;
```

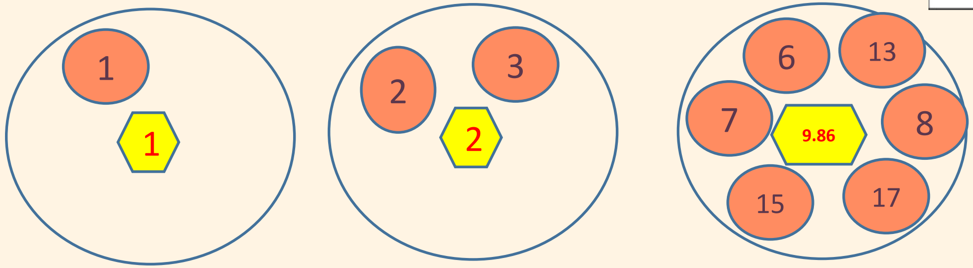
Example

- **Dataset**



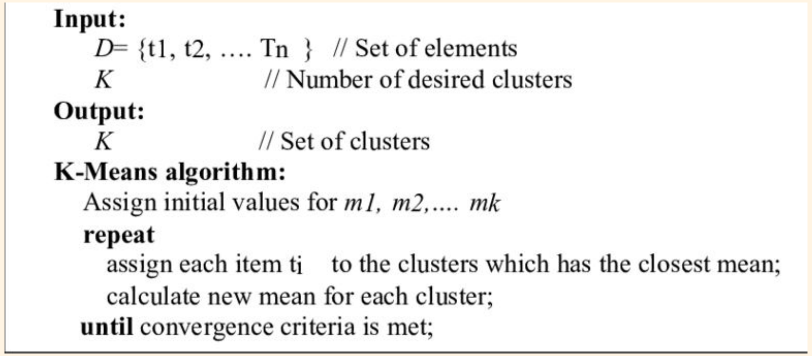
```
Input:
    D= {t1, t2, …. Tn }   // Set of elements
    K                     // Number of desired clusters
Output:
    K                     // Set of clusters
K-Means algorithm:
  Assign initial values for m1, m2,…. mk
  repeat
    assign each item ti   to the clusters which has the closest mean;
    calculate new mean for each cluster;
  until convergence criteria is met;
```

- **Reassign each object to its closest cluster**



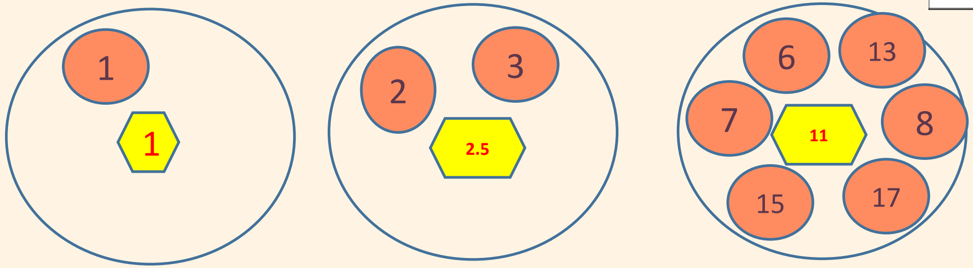
Example

- **Dataset**



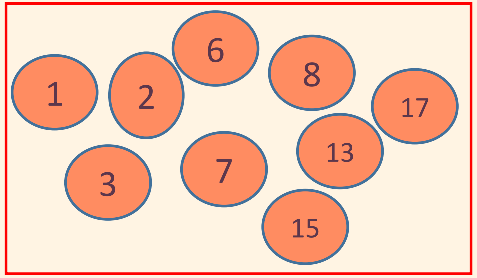
```
Input:
    D= {t1, t2, .... Tn }   // Set of elements
    K                       // Number of desired clusters
Output:
    K                       // Set of clusters
K-Means algorithm:
  Assign initial values for m1, m2,.... mk
  repeat
      assign each item ti  to the clusters which has the closest mean;
      calculate new mean for each cluster;
  until convergence criteria is met;
```

- **Recalculate the center**

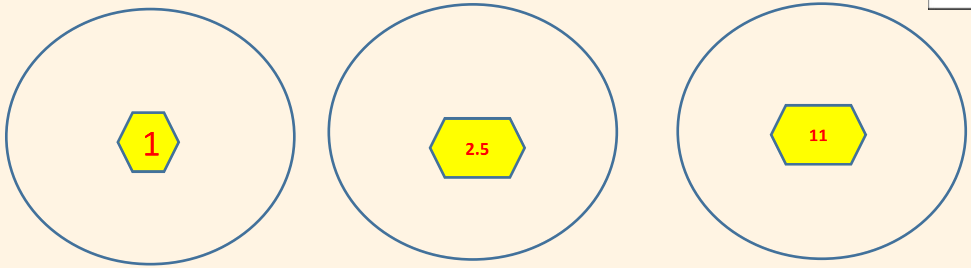


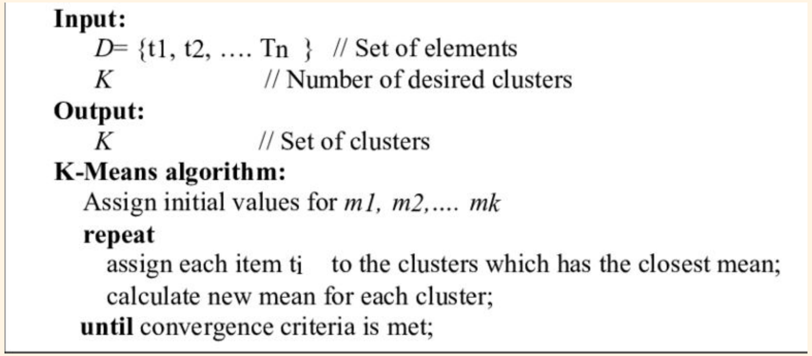
Example

- **Dataset**



- **Reassign each object to its closest cluster**

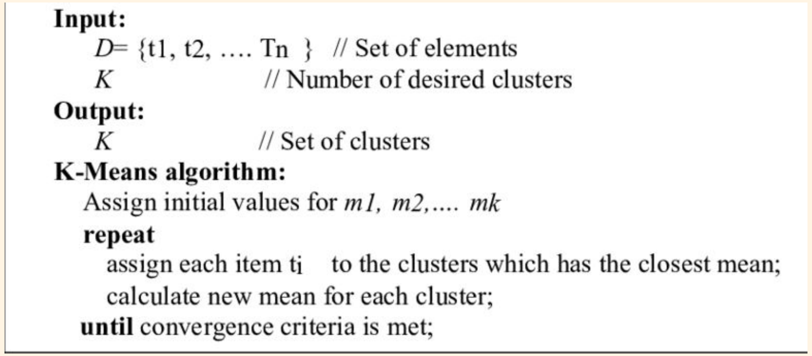




```
Input:
    D= {t1, t2, .... Tn }   // Set of elements
    K                       // Number of desired clusters
Output:
    K                       // Set of clusters
K-Means algorithm:
  Assign initial values for m1, m2,.... mk
  repeat
    assign each item $t_i$ to the clusters which has the closest mean;
    calculate new mean for each cluster;
  until convergence criteria is met;
```

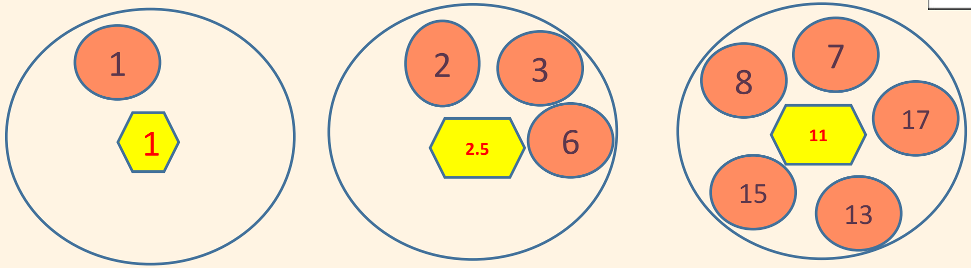
Example

- Dataset



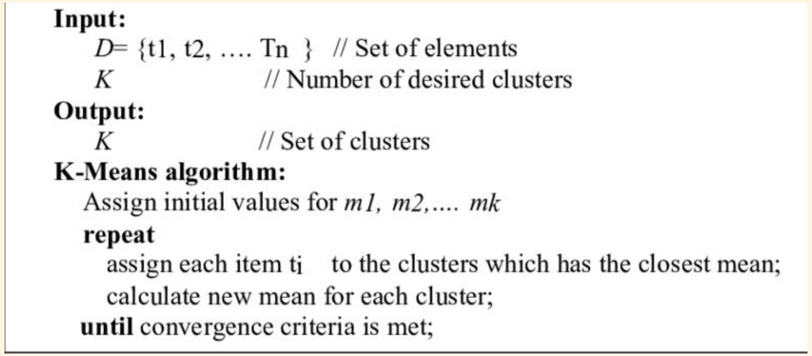
```
Input:
    D= {t1, t2, .... Tn }   // Set of elements
    K                       // Number of desired clusters
Output:
    K                       // Set of clusters
K-Means algorithm:
  Assign initial values for m1, m2,.... mk
  repeat
    assign each item ti  to the clusters which has the closest mean;
    calculate new mean for each cluster;
  until convergence criteria is met;
```

- Reassign each object to its closest cluster



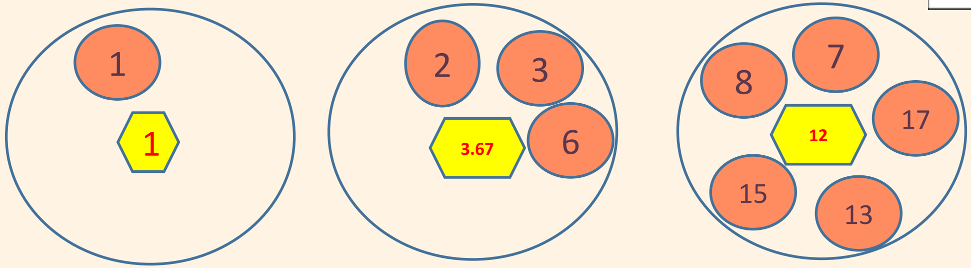
Example

- **Dataset**



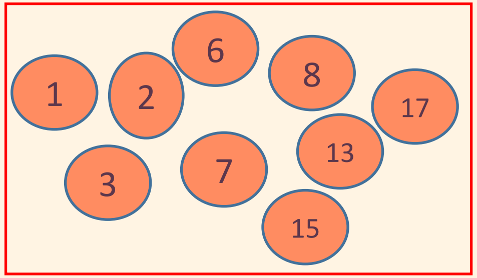
```
Input:
    D= {t1, t2, .... Tn }  // Set of elements
    K                      // Number of desired clusters
Output:
    K                      // Set of clusters
K-Means algorithm:
   Assign initial values for m1, m2,.... mk
   repeat
      assign each item ti   to the clusters which has the closest mean;
      calculate new mean for each cluster;
   until convergence criteria is met;
```

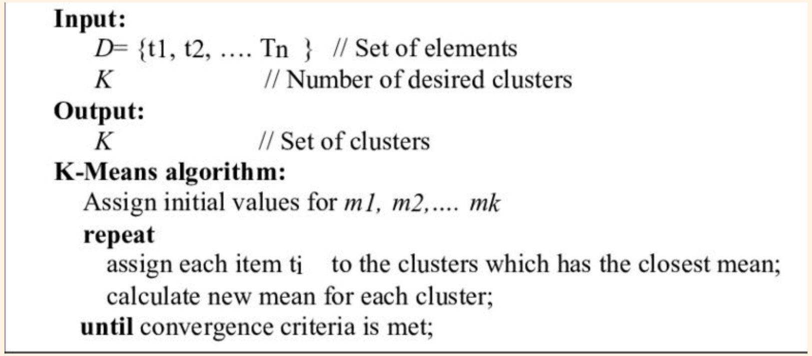
- **Recalculate the center**



Example

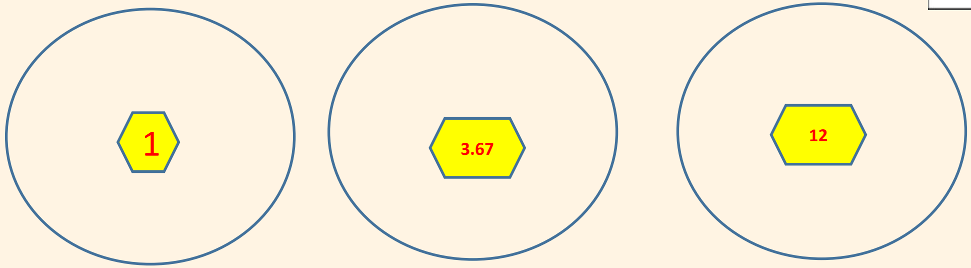
- **Dataset**





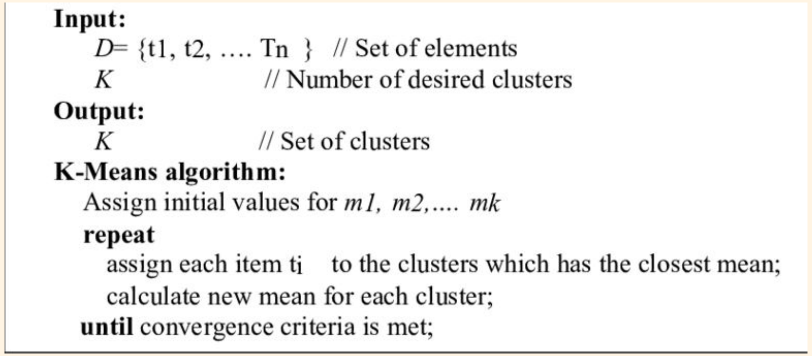
```
Input:
    D= {t1, t2, …. Tn }   // Set of elements
    K                     // Number of desired clusters
Output:
    K                     // Set of clusters
K-Means algorithm:
  Assign initial values for m1, m2,…. mk
  repeat
    assign each item ti  to the clusters which has the closest mean;
    calculate new mean for each cluster;
  until convergence criteria is met;
```

- **Reassign each object to its closest cluster**



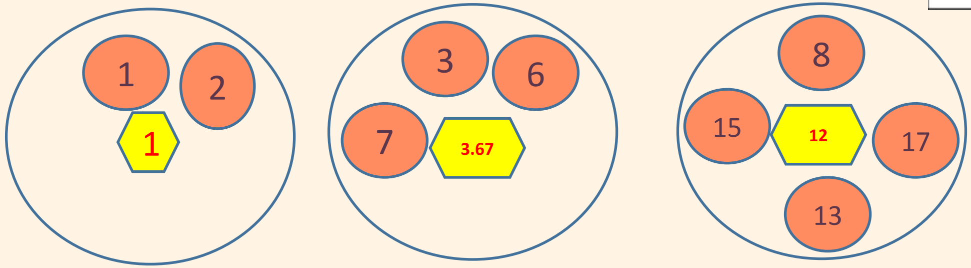
Example

- **Dataset**



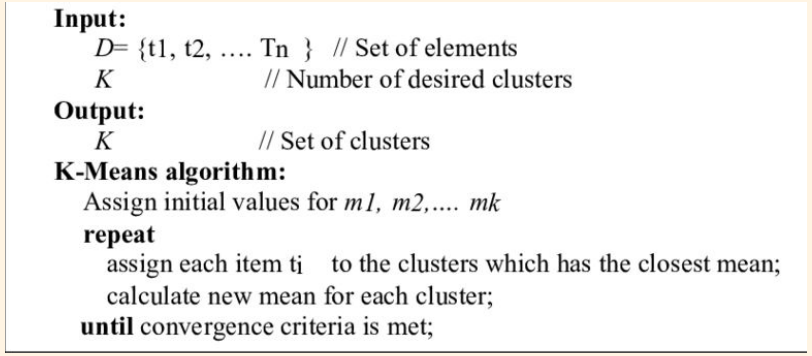
```
Input:
    D= {t1, t2, .... Tn }   // Set of elements
    K                       // Number of desired clusters
Output:
    K                       // Set of clusters
K-Means algorithm:
  Assign initial values for m1, m2,.... mk
  repeat
    assign each item ti   to the clusters which has the closest mean;
    calculate new mean for each cluster;
  until convergence criteria is met;
```

- **Reassign each object to its closest cluster**



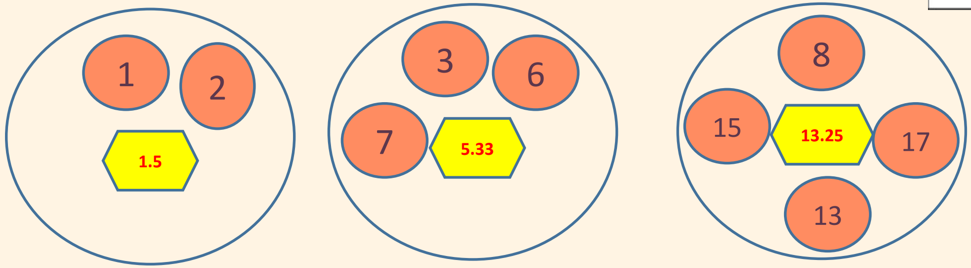
Exemple

- **Dataset**



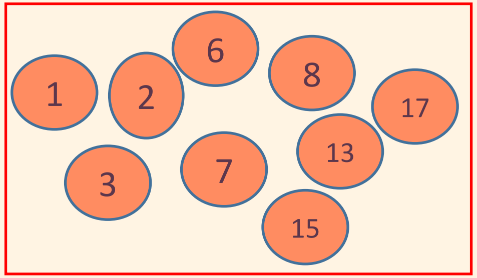
```
Input:
    D= {t1, t2, .... Tn }   // Set of elements
    K                       // Number of desired clusters
Output:
    K                       // Set of clusters
K-Means algorithm:
   Assign initial values for m1, m2,.... mk
   repeat
      assign each item ti   to the clusters which has the closest mean;
      calculate new mean for each cluster;
   until convergence criteria is met;
```

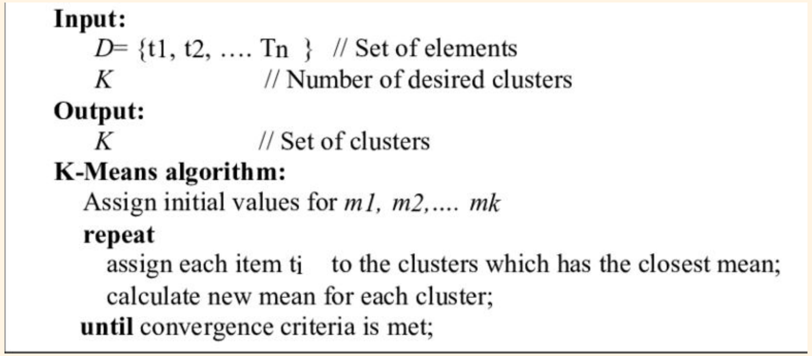
- **Recalculate the center**



Example

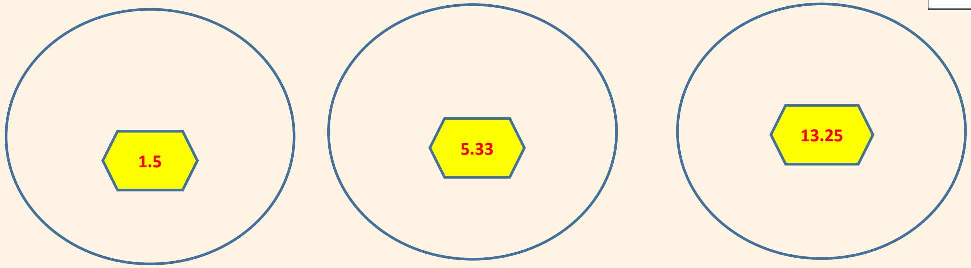
- **Dataset**





```
Input:
    D= {t1, t2, .... Tn }   // Set of elements
    K                       // Number of desired clusters
Output:
    K                       // Set of clusters
K-Means algorithm:
  Assign initial values for m1, m2,.... mk
  repeat
      assign each item ti   to the clusters which has the closest mean;
      calculate new mean for each cluster;
  until convergence criteria is met;
```

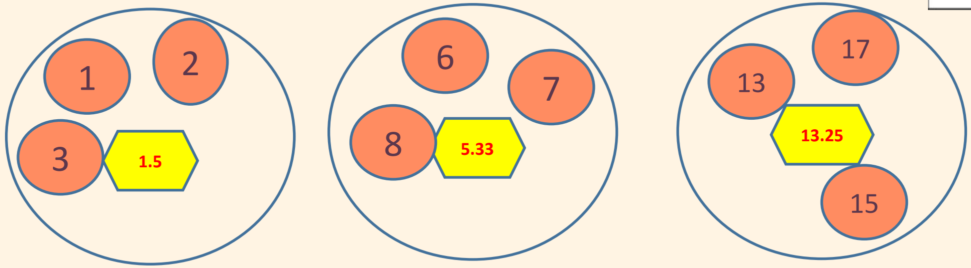
- **Reassign each object to its closest cluster**

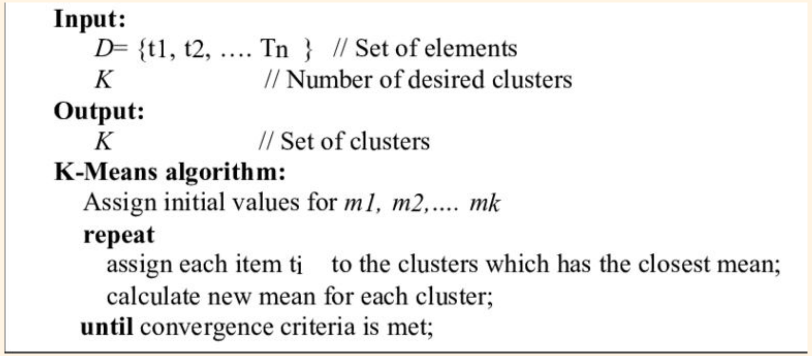


Example

- **Dataset**

- **Reassign each object to its closest cluster**



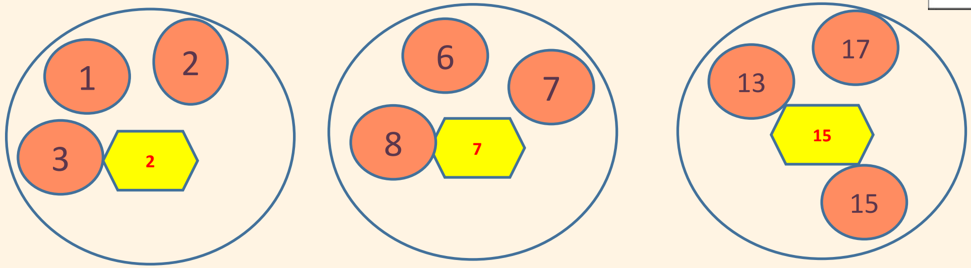


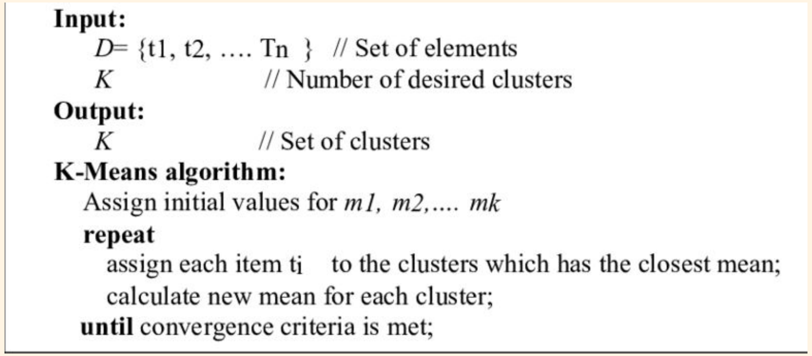
```
Input:
    D= {t1, t2, .... Tn }   // Set of elements
    K                        // Number of desired clusters
Output:
    K                        // Set of clusters
K-Means algorithm:
  Assign initial values for m1, m2,.... mk
  repeat
     assign each item ti   to the clusters which has the closest mean;
     calculate new mean for each cluster;
  until convergence criteria is met;
```

Example

- **Dataset**

- **Recalculate the center**

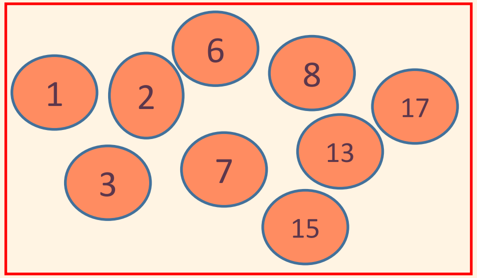




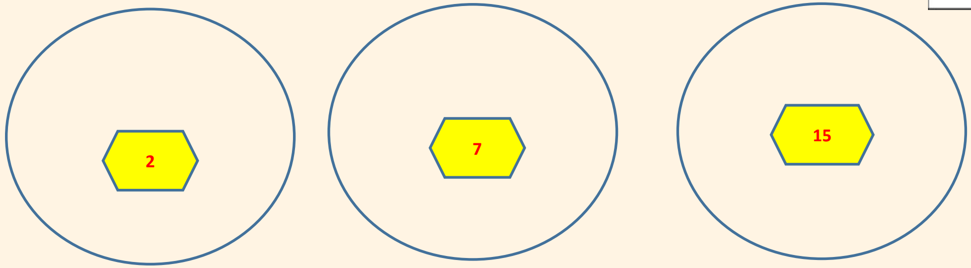
```
Input:
    D= {t1, t2, .... Tn }   // Set of elements
    K                       // Number of desired clusters
Output:
    K                       // Set of clusters
K-Means algorithm:
  Assign initial values for m1, m2,.... mk
  repeat
    assign each item ti   to the clusters which has the closest mean;
    calculate new mean for each cluster;
  until convergence criteria is met;
```

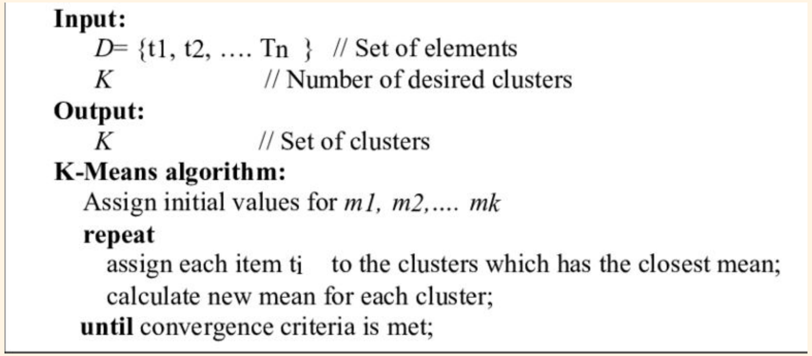
Example

- **Dataset**



- **Reassign each object to its closest cluster**

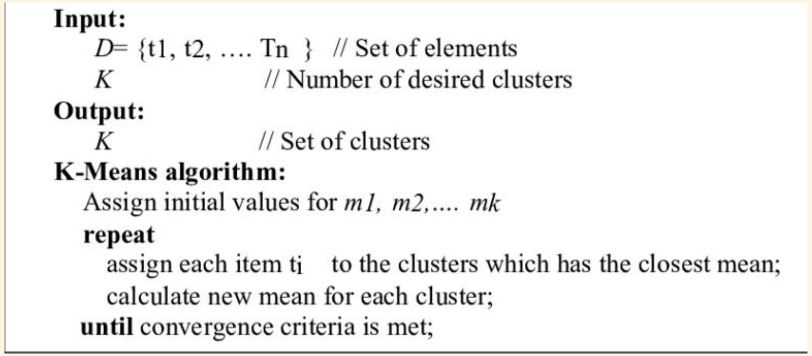




```
Input:
    D= {t1, t2, .... Tn }  // Set of elements
    K                      // Number of desired clusters
Output:
    K                      // Set of clusters
K-Means algorithm:
  Assign initial values for m1, m2,.... mk
  repeat
    assign each item ti  to the clusters which has the closest mean;
    calculate new mean for each cluster;
  until convergence criteria is met;
```

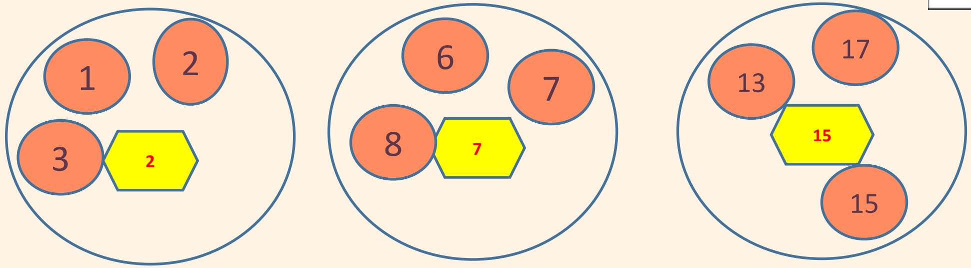
Example

- **Dataset**



```
Input:
    D= {t1, t2, .... Tn }   // Set of elements
    K                  // Number of desired clusters
Output:
    K                  // Set of clusters
K-Means algorithm:
   Assign initial values for m1, m2,.... mk
   repeat
      assign each item ti   to the clusters which has the closest mean;
      calculate new mean for each cluster;
   until convergence criteria is met;
```

- **Recalculate the center**



- **No changement of the centroid**

Thanks