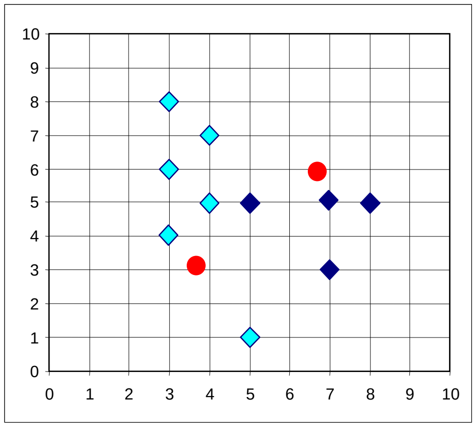
RADIAL-BASIS FUNCTION NETWORKS

INT301 Bio-computation, Week 9, 2021

K-Means Clustering Method

- Choose the number of clusters - K
- Randomly choose initial positions of K centroids
- Assign each of the points to the "nearest centroid" (depends on distance measure)
- Re-compute centroid positions
- If solution converges → Stop!

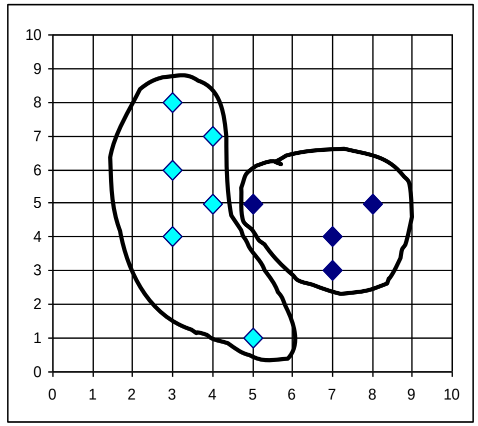
K-Means Clustering Method



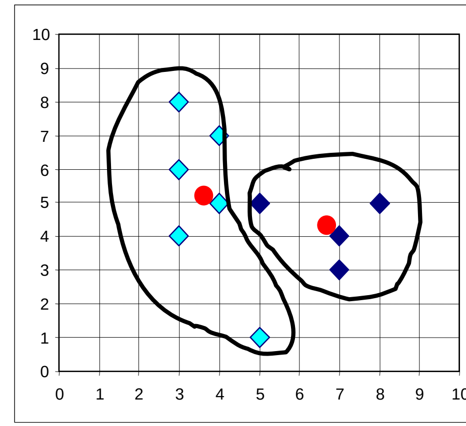
K=2

Arbitrarily choose K object as initial cluster center

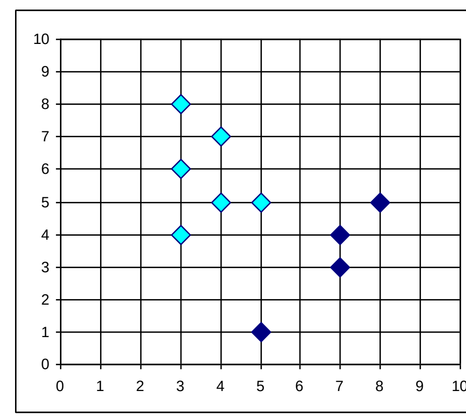
Assign each objects to most similar center



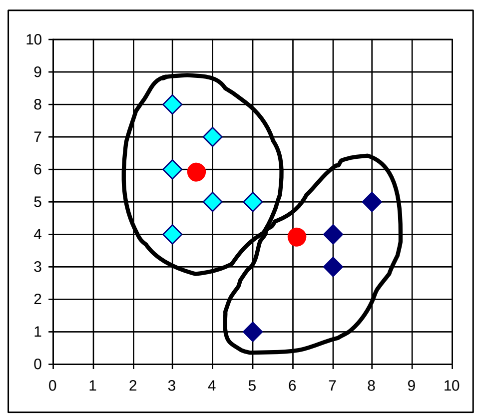
Update the cluster means



reassign



Update the cluster means



reassign

A Simple Example

Individual	Variable 1	Variable 2
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

A Simple Example

Step 1:

Initialization: Randomly we choose following two centroids (k=2) for two clusters.

In this case the 2 centroid are: m1=(1.0,1.0) and m2=(5.0,7.0).

Individual	Variable 1	Variable 2
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

	Individual	Mean Vector
Group 1	1	(1.0, 1.0)
Group 2	4	(5.0, 7.0)

A Simple Example

Step 2:

- From Euclidean distance computation, we obtain two clusters containing: {1,2,3} and {4,5,6,7}.

$$d(m_1,2)=\sqrt{|1.0-1.5|^2+|1.0-2.0|^2}=1.12$$

$$d(m_2,2)=\sqrt{|5.0-1.5|^2+|7.0-2.0|^2}=6.10$$

- Their new centroids are:

Individual	Centroid 1 (1.0, 1.0)	Centroid 2 (5.0, 7.0)
1 (1.0, 1.0)	0	7.21
2 (1.5, 2.0)	1.12	6.10
3 (3.0, 4.0)	3.61	3.61
4 (5.0, 7.0)	7.21	0
5 (3.5, 5.0)	4.72	2.5
6 (4.5, 5.0)	5.31	2.06
7 (3.5, 4.5)	4.30	2.92

$$m_1=\left(\frac{1}{3}(1.0+1.5+3.0),\frac{1}{3}(1.0+2.0+4.0)\right)=(1.83,2.33)$$

$$m_2=\left(\frac{1}{4}(5.0+3.5+4.5+3.5),\frac{1}{4}(7.0+5.0+5.0+4.5)\right)=(4.12,5.38)$$

A Simple Example

Step 3:

- Now using these centroids we compute the Euclidean distance of each object
- Therefore, the new clusters are:

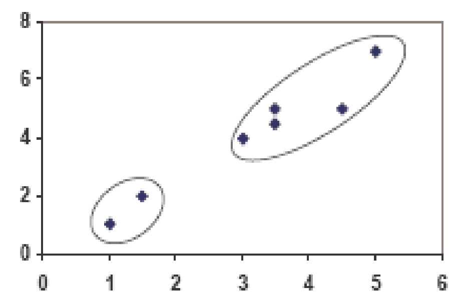
  {1,2} and {**3**,4,5,6,7}
- Next centroids are: m1=(1.25, 1.5) and m2=(3.9, 5.1)

Individual	Centroid 1 (1.83, 2.33)	Centroid 2 (4.12, 5.38)
1 (1.0, 1.0)	1.57	5.38
2 (1.5, 2.0)	0.47	4.28
3 (3.0, 4.0)	2.04	1.78
4 (5.0, 7.0)	5.64	1.84
5 (3.5, 5.0)	3.15	0.73
6 (4.5, 5.0)	3.78	0.54
7 (3.5, 4.5)	2.74	1.08

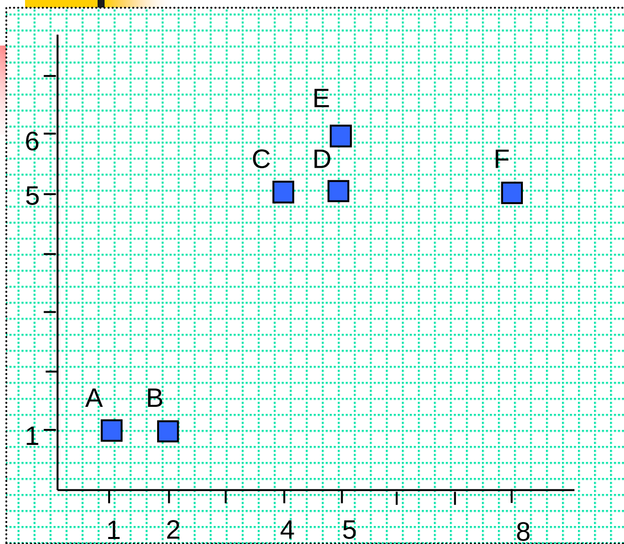
A Simple Example

- Step 4 :
  The clusters obtained are:
  {1,2} and {3,4,5,6,7}
- Therefore, there is no change in the cluster.
- Thus, the algorithm comes to a halt here and final result consist of 2 clusters {1,2} and {3,4,5,6,7}.

Individual	Centroid 1 (1.25, 1.5)	Centroid 2 (3.9, 5.1)
1 (1.0, 1.0)	0.56	5.02
2 (1.5, 2.0)	0.56	3.92
3 (3.0, 4.0)	3.05	1.42
4 (5.0, 7.0)	6.66	2.20
5 (3.5, 5.0)	4.16	0.41
6 (4.5, 5.0)	4.78	0.61
7 (3.5, 4.5)	3.75	0.72

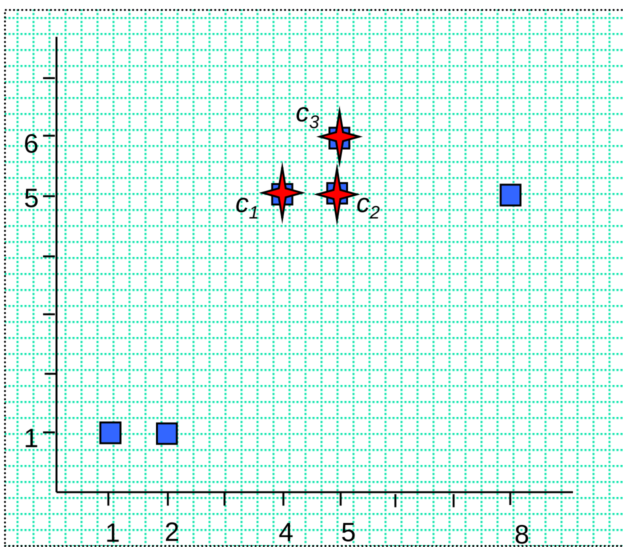


K-means example 2



Data set

X: a set of N data vectors
$N = 6$

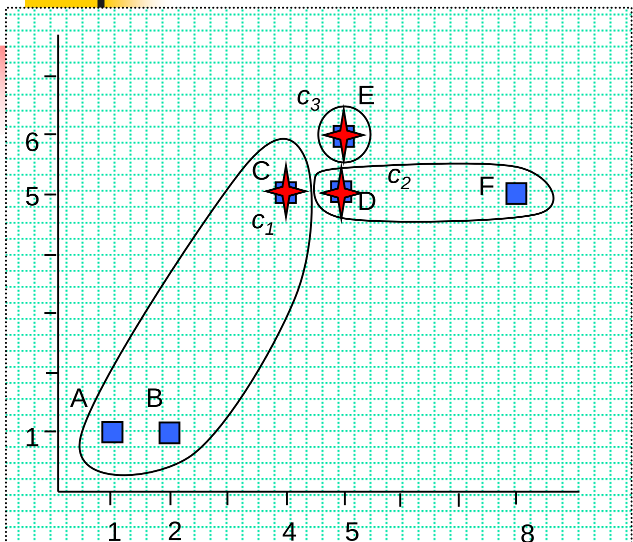


Number of clusters
Random initial centroids

Initial codebook:
$c_1 = C,\ c_2 = D,\ c_3 = E$

C_I: initialized k cluster centroids
$k = 3$

K-means example 2

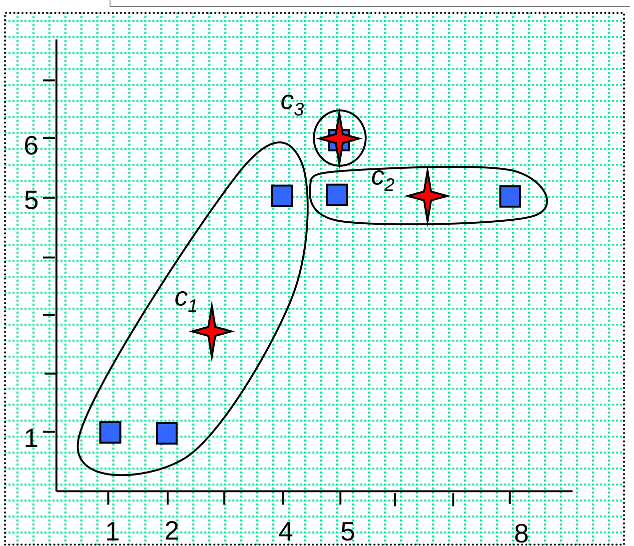


Generate optimal partitions

Distance matrix (Euclidean distance)

$$
\begin{array}{c c}
 & \begin{array}{cccccc} \mathbf{A} & \mathbf{B} & \mathbf{C} & \mathbf{D} & \mathbf{E} & \mathbf{F} \end{array} \\
\begin{array}{c} c_1 \\ c_2 \\ c_3 \end{array} & \begin{pmatrix} 5 & 4.5 & 0 & 1 & 1.4 & 4 \\ 5.7 & 5 & 1 & 0 & 1 & 3 \\ 6.4 & 5.8 & 1.4 & 1 & 0 & 3.2 \end{pmatrix}
\end{array}
$$

After 1st iteration: MSE = 9.0



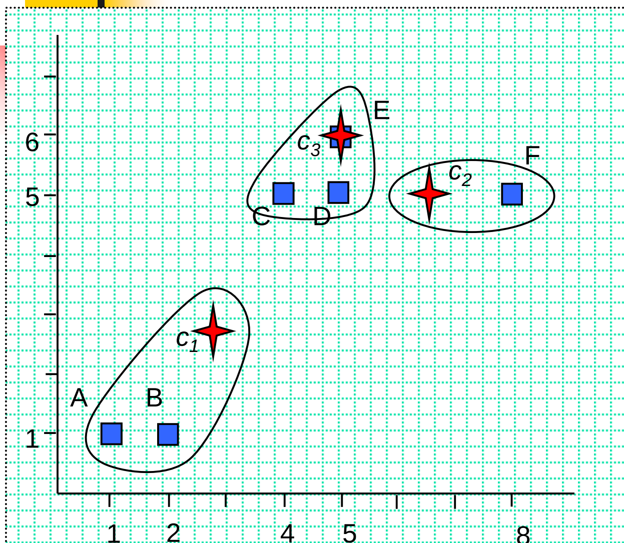
Generate optimal centroids

$$c_1 = \left(\frac{1+2+4}{3}, \frac{1+1+5}{3}\right) = (2.3, 2.3)$$

$$c_2 = \left(\frac{5+8}{2}, \frac{5+5}{2}\right) = (6.5, 5)$$

$$c_3 = (5, 6)$$

K-means example 2

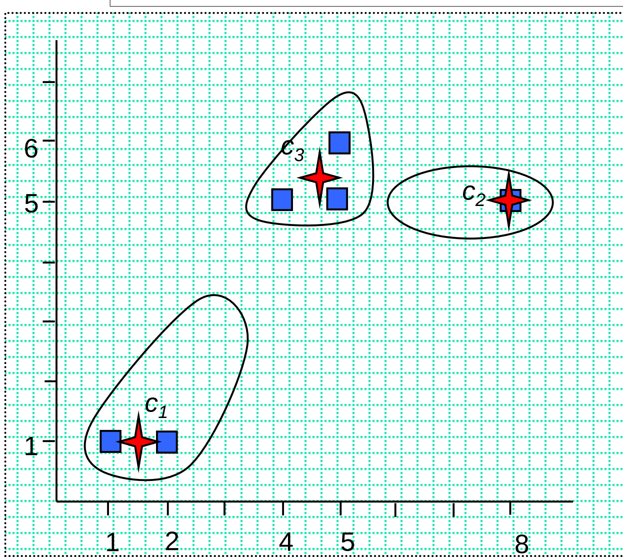


Generate optimal partitions

Distance matrix (Euclidean distance)

$$\begin{array}{c} \\ c_1 \\ c_2 \\ c_3 \end{array} \begin{array}{c} \begin{array}{cccccc} \mathbf{A} & \mathbf{B} & \mathbf{C} & \mathbf{D} & \mathbf{E} & \mathbf{F} \end{array} \\ \begin{pmatrix} 1.9 & 1.4 & 3.1 & 3.8 & 4.5 & 6.3 \\ 6.8 & 6 & 2.5 & 1.5 & 1.8 & 1.5 \\ 6.4 & 5.8 & 1.4 & 1 & 0 & 3.2 \end{pmatrix} \end{array}$$

After 2nd iteration: MSE = 1.78



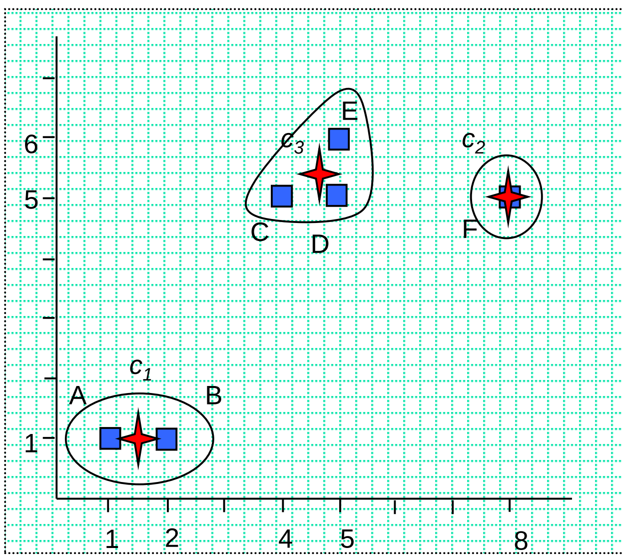
Generate optimal centroids

$$c_1 = \left(\frac{1+2}{2}, \frac{1+1}{2} \right) = (1.5,1)$$

$$c_2 = (8,5)$$

$$c_3 = \left(\frac{4+5+5}{3}, \frac{5+5+6}{3} \right) = (4.7,5.3)$$

K-means example 2



Generate optimal partitions

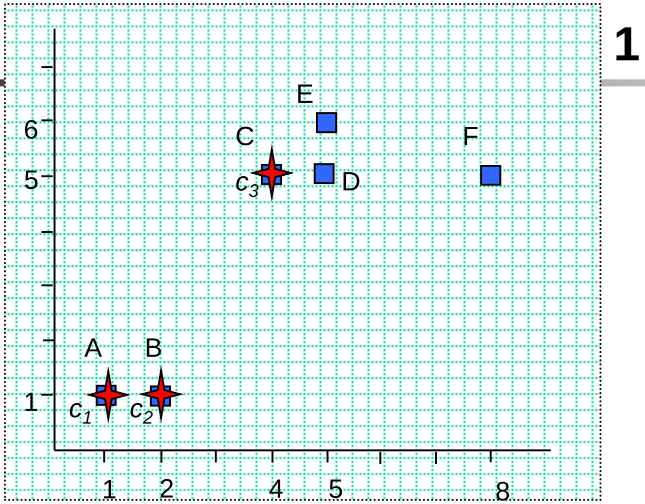
Distance matrix (Euclidean distance)

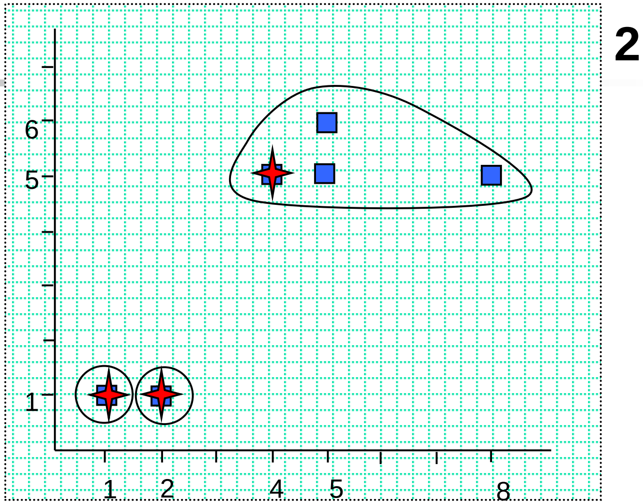
$$
\begin{array}{c c}
 & \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\
\begin{array}{c} c_1 \\ c_2 \\ c_3 \end{array} &
\begin{pmatrix}
0.5 & 0.5 & 4.7 & 5.3 & 6.1 & 7.6 \\
8.1 & 7.2 & 4 & 3 & 3.2 & 0 \\
5.7 & 5.1 & 0.7 & 0.5 & 0.7 & 3.3
\end{pmatrix}
\end{array}
$$

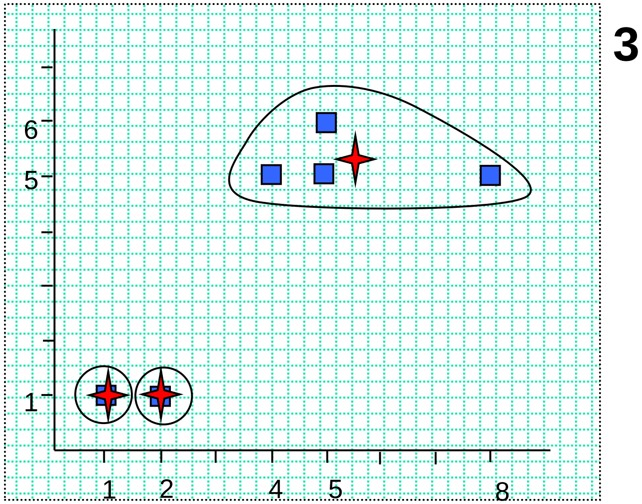
After 3rd iteration: MSE = 0.31

No object move - stop

K-means example 2







Initial codebook:

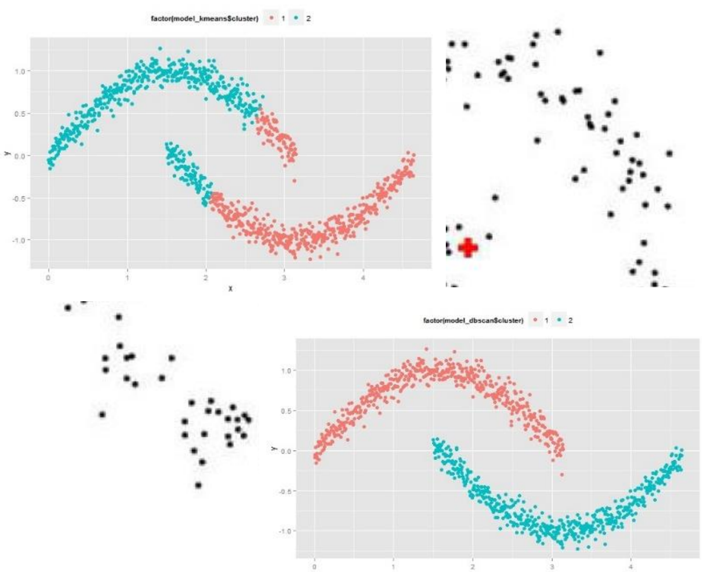
$c_1 = A, c_2 = B, c_3 = C$

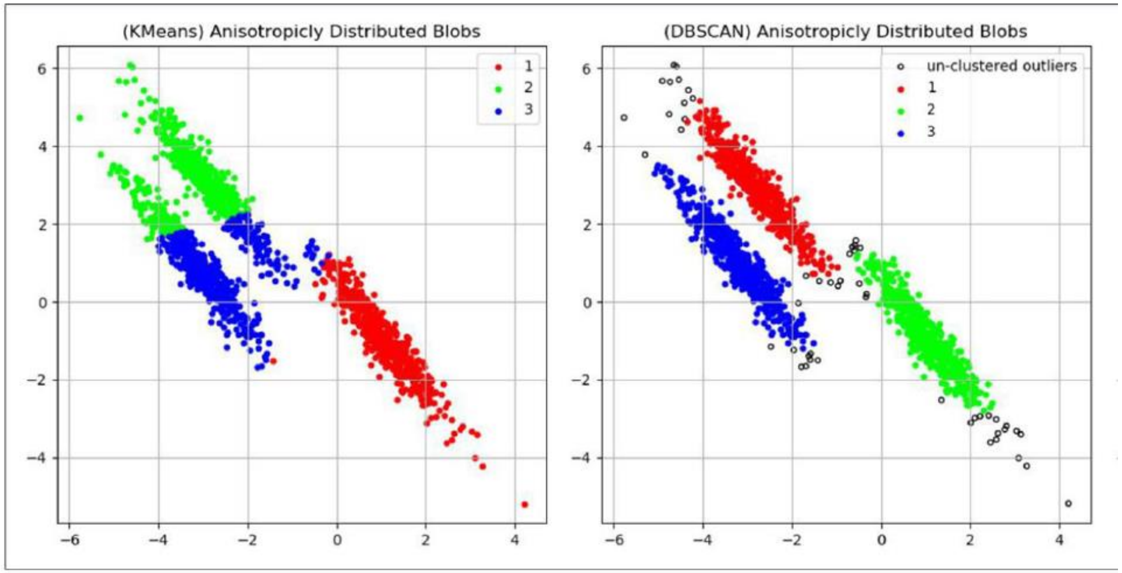
Issues

- The numbers of clusters must be specified in advance
- This method is not suitable for clusters with non-convex shapes
- This method is sensitive to noise and outlier elements

Issues

- This method is not suitable for clusters with non-convex shapes
- This method is sensitive to noise and outlier elements





Two ways to improve k-means

- **Repeated k-means**
  - Try several random initializations and take the best.

- **Better initialization**
  - Use some better heuristic to allocate the initial distribution of code vectors.
    - *Designing good initialization is not any easier than designing good clustering algorithm at the first place!*

RBF Networks vs. MLP

- **Similarities**
  - The RBF Networks and the MLP are layered feedforward networks that produce nonlinear function mappings;
  - They are both proven to be universal approximators;
- **Differences**
  - An RBF network has only one hidden layer, while MLP networks have one or more hidden layers depending on the application task;
  - The nodes in the hidden and output layers of MLP use the same activation function, while RBF uses different activation functions at each node (Gaussians parameterized by different centers and variances);
  - The hidden and output layers of MLP are both nonlinear, while only the hidden layer of RBF is nonlinear (the output layer is linear);
  - The activation functions in the RBF nodes compute the Euclidean distance between the input examples and the centers, while the activation functions of MLP compute inner products from the input examples and the incoming weights;
  - MLP constructs global approximations while RBF construct local approximations.

RBF Training Algorithm

- **Initialization:** Examples $\{(\mathbf{x}_e, d_e)\}_{e=1}^{N}$
- **Determine:**
  - the network structure with a number n of basis functions φ_i, i=1,2,...,n
  - the basis function centers $\mathbf{t}_i$, i=1,2,...,n
    - e.g.: using ***k-means clustering*** algorithm
  - the basis function variances σ_i^2, i=1,2,...,n
- **Compute:**
  - the outputs from each example with the Gaussian basis functions:

$$\phi_{ei} = \frac{\exp(-||x_e - t_i||^2)}{2\sigma_i^2}$$

  - the weights: $W = (\Phi^{\mathrm{T}}\Phi)^{-1}\Phi^T d$

Implementing RBF Networks

XOR Example

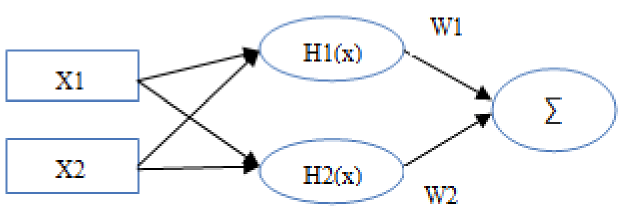
Consider the XOR example:

(x_1,x_2) | Y

(0, 0) | 0
(0, 1) | 1
(1, 0) | 1
(1, 1) | 0

- The XOR example can be learned by an RBF network with the following structure
  - two hidden nodes
  - one output node

$$F(x) = \sum_{i=1}^{2} w_i \exp\left(-\frac{||x - t_i||^2}{2{\sigma_i}^2}\right)$$



Implementing RBF Networks

- **Solution**: Training of this RBF to learn the XOR examples involves the following steps

- **Initialization**:
  - the network structure has two basis functions φ_i

    $\varphi_i = \exp(-||\mathbf{x} - \mathbf{x}_i||^2)$
  - choose the centers: $\mathbf{x}_1 = (1,1)$, $\mathbf{x}_2 = (0,0)$, $2\sigma_i^2 = 1$

XOR

(x_1,x_2) | Y

(0, 0) | 0
(0, 1) | 1
(1, 0) | 1
(1, 1) | 0

Implementing RBF Networks

- **Training:**

φ_{11} = exp(-((0-1)^2+(0-1)^2)) = exp(-2) = 0.1353
φ_{21} = exp(-((0-1)^2+(1-1)^2)) = exp(-1) = 0.3678
φ_{31} = exp(-((1-1)^2+(0-1)^2)) = exp(-1) = 0.3678
φ_{41} = exp(-((1-1)^2+(1-1)^2)) = exp(0) = 1
φ_{12} = exp(-((0-0)^2+(0-0)^2)) = exp(0) = 1
φ_{22} = exp(-((0-0)^2+(1-0)^2)) = exp(-1) = 0.3678
φ_{32} = exp(-((1-0)^2+(0-0)^2)) = exp(-1) = 0.3678

$$\Phi = \begin{bmatrix} \phi_{11} & \phi_{12} & \cdots & \phi_{1n} \\ \phi_{21} & \phi_{22} & \cdots & \phi_{2n} \\ & & & \\ \phi_{N1} & \phi_{N2} & \cdots & \phi_{Nn} \end{bmatrix} \qquad \Phi = \begin{bmatrix} \phi_{11} & \phi_{12} \\ \phi_{21} & \phi_{22} \\ \phi_{31} & \phi_{32} \\ \phi_{41} & \phi_{42} \end{bmatrix} = \begin{bmatrix} 0.1353 & 1 \\ 0.3678 & 0.3678 \\ 0.3678 & 0.3678 \\ 1 & 0.1353 \end{bmatrix}$$

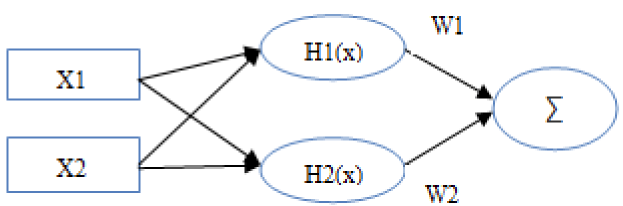
Implementing RBF Networks

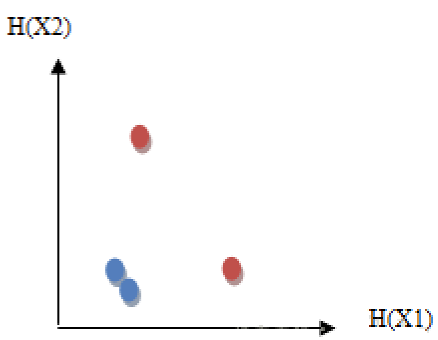
$$
\Phi = \begin{bmatrix} \phi_{11} & \phi_{12} \\ \phi_{21} & \phi_{22} \\ \phi_{31} & \phi_{32} \\ \phi_{41} & \phi_{42} \end{bmatrix} = \begin{array}{cc} H1(x) & H2(x) \\ \left[\begin{array}{cc} \boxed{0.1353 \quad 1} \\ 0.3678 & 0.3678 \\ 0.3678 & 0.3678 \\ \boxed{1 \quad 0.1353} \end{array}\right] \end{array}
$$

XOR

(x_1, x_2) | Y

(0, 0) | 0
(0, 1) | 1
(1, 0) | 1
(1, 1) | 0





THANK YOU

VISIT US

WWW.XJTLU.EDU.CN

FOLLOW US

@XJTLU

Xi'an Jiaotong-Liverpool University
西交利物浦大学