RADIAL BASIS FUNCTION NETWORKS
MODULE 5

Dr. Rajesh B. Ghongade

Professor, Vishwakarma Institute of Information Technology, Pune-411048

Agenda

- Concept of RBFN and Cover's Theorem
- RBFN architecture
- ϕ functions
- XOR problem
- RBFN Algorithm
- K-means clustering
- MATLAB Demo
- Comparison between MLP and RBFN
- Applications of RBFN

Concept of RBFN

- MLP can be viewed as a stochastic approximation approach but we can view it also as a surface fitting consideration
- MLP succeeds because of the in-between mapping done by the hidden layer
- For a problem where we cannot pass a hyper plane for separation of the classes i.e. non-linearly separable problems, but we may be able to pass a hyper sphere or a hyper quadric (multi-dimensional) surface
- Thus given a non linearly separable problem our task is to determine a hyper surface that can classify properly, which is a surface fitting problem in multi dimensional space and is the essence of RBF networks
- **RBF = radial-basis function: a function which depends only on the radial distance from a point**

Cover's Theorem

" ***A pattern classification problem cast in a high dimensional space is more likely to be linearly separable than in low dimensional space***"

Consider a set H of N patterns $\overrightarrow{X_1}, \overrightarrow{X_2}, \ldots, \overrightarrow{X_N}$

Each vector can be assigned to any of the two classes H_1 *or* H_2

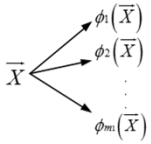
If we have a family of surfaces wherein at least one surface separates the two classes then we say that the set H is separable with respect to that family of surfaces

$\vec{X}$ has dimensions= m_0

For each $\vec{X} \in H$, define a vector made of a set of real valued functions

$\left\{\phi\left(\vec{X}\right) \middle| i = 1, 2, ..., m_1\right\}$ .... m_1 number of real value functions

These are the hidden layer functions so that input vector gets transformed as:



$\therefore$ Space that is spanned by the set of hidden functions is:

$\left\{\phi_i\left(\vec{X}\right)\right\}_{i=1}^{m_1}$ is the hidden space or feature space

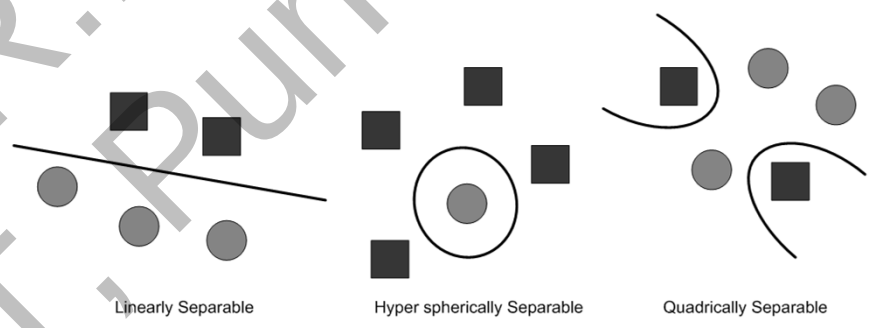
We are mapping m_0 dimension inputs into m_1 dimension hidden space

A dichotomy $\{H_1, H_2\}$ is ϕ-separable if there exists one m_1 dimensional vector such that:

$\vec{W^T} \cdot \phi\left(\vec{X}\right) > 0$ then $\vec{X} \in H_1$

$\vec{W^T} \cdot \phi\left(\vec{X}\right) < 0$ then $\vec{X} \in H_2$

Hyper plane equation is: $\vec{W^T} \cdot \phi\left(\vec{X}\right) = 0$ is the separating surface in ϕ space



If $\vec{X_1}, \vec{X_2}, ..., \vec{X_N}$ are independently chosen patterns and there are large number of dichotomies, but only a few are separable,

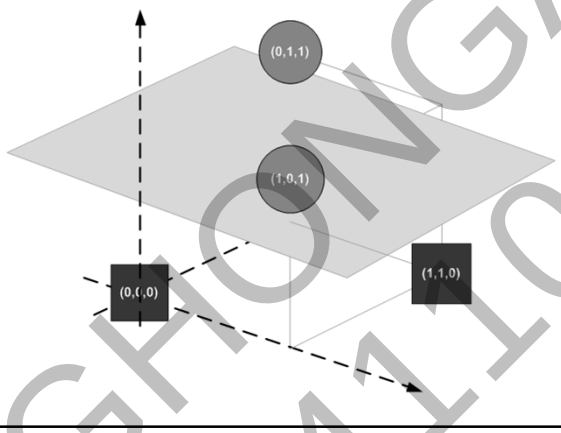
Cover studied the probabilities of those dichotomies which are separable

Assuming all possible dichotomies of $H = \left\{\vec{X}\right\}_{i=1}^{N}$ are equiprobable

The probability $P(N, m_1)$ denotes the probability that a particular dichotomy picked at random is ϕ – separable is given by Cover's theorem as:

$$P(N, m_1) = \left(\frac{1}{2}\right)^{N-1} \sum_{m=0}^{m_i-1} \binom{N-1}{m}$$

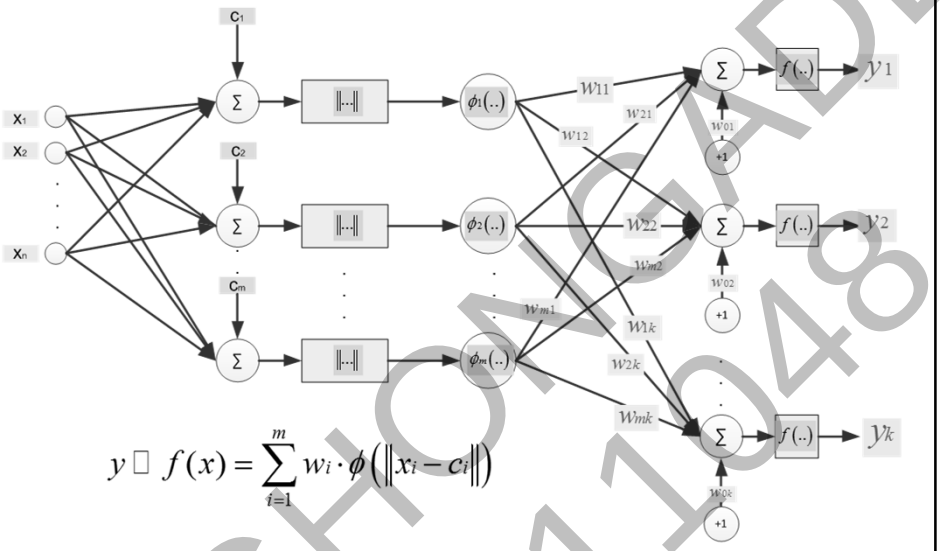
If m_i is very high the probability is closer to 1



RBFN Architecture

- RBFNs have hidden units which provide a set of functions which forms a basis for mapping into the hidden layer space, hence the need for some basis functions
- Hidden neurons provide the basis functions hence the name radial basis function networks
- Basic form of RBFN consists of three layers:
  - Input layer contains source nodes connected to the environment
  - Only one hidden layer which does the non-linear transformation (mapping) from input space to hidden space, where hidden space dimensionality is higher than input space
  - Output layer supplies the response

A typical RBF Network

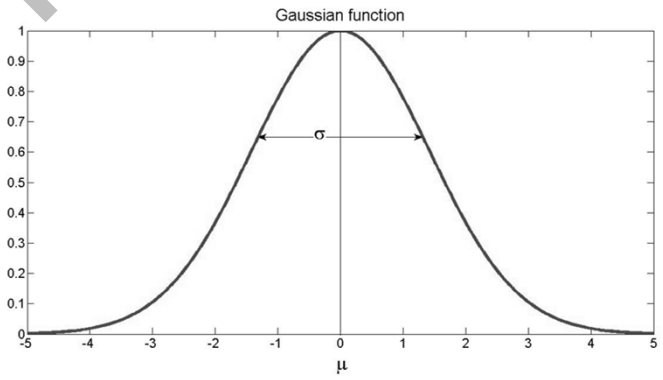


$$y = f(x) = \sum_{i=1}^{m} w_i \cdot \phi\left(\|x_i - c_i\|\right)$$

ϕ functions

1) Gaussian

$$\phi(r) = \exp\left(-\frac{r^2}{2\sigma^2}\right) \quad \text{for some } \sigma > 0, r \in R$$

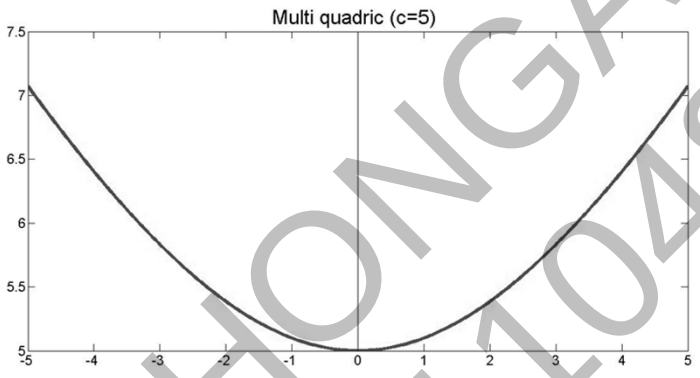


Localized function ; if r→ ∞, $\emptyset(r) \to 0$

ϕ functions

2) Multi-quadrics

$$\phi(r) = \left(r^2 + c^2\right)^{1/2} \quad \text{for some } c > 0, r \in R$$

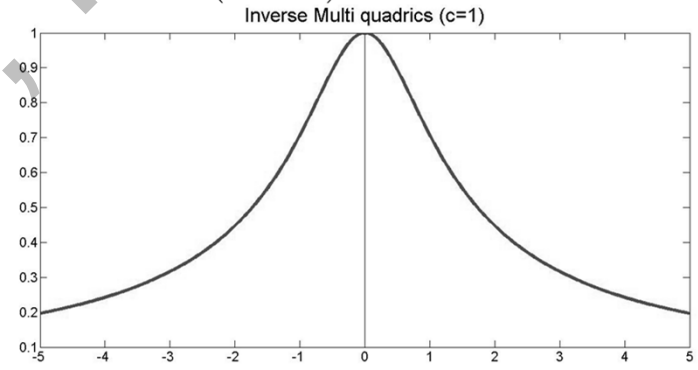


Non-localized function ; if r→ ∞, $\emptyset(r) \to \infty$

ϕ functions

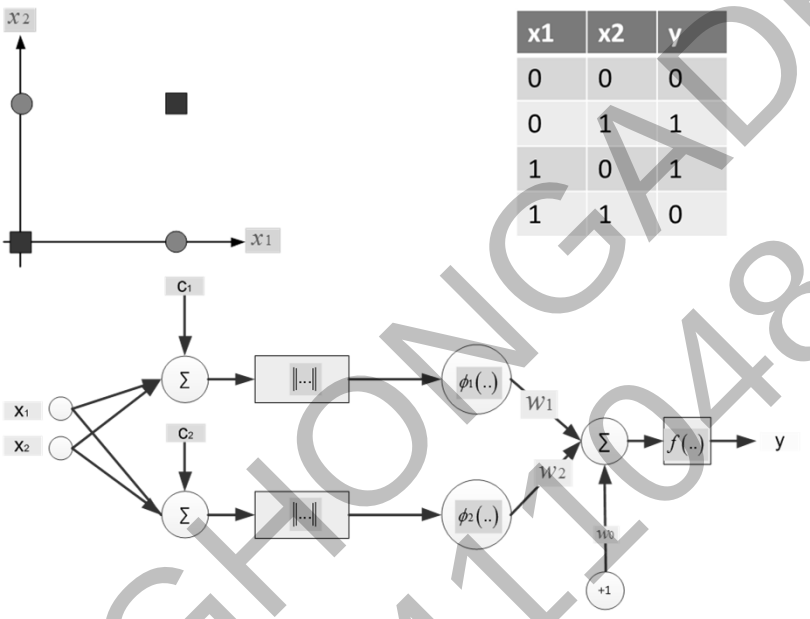
3) Inverse Multi-quadrics

$$\phi(r) = \frac{1}{\left(r^2 + c^2\right)^{1/2}} \quad \text{for some } c > 0, r \in R$$



Localized function ; if r→ ∞, $\emptyset(r) \to 0$

XOR problem



x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0

We select the two φ-functions as Gaussian with μ=0 and σ= $\sqrt{1/2}$ so that

$$\phi(r) = \exp\left(-\frac{r^2}{2\sigma^2}\right) \quad \text{reduces to} \quad \phi(r) = \exp\left(-r^2\right)$$

We select the centers as C_1=[0,0] and C_2 =[1,1] hence:

$$\phi_1\left(\vec{X}\right) = \exp\left(-\left\|\vec{X} - C_1\right\|^2\right)$$

$$\phi_2\left(\vec{X}\right) = \exp\left(-\left\|\vec{X} - C_2\right\|^2\right)$$

Thus the inputs are x, the distances from the centers are r, and the outputs from the hidden units are ∅

When all the inputs are presented to the net we get:

For $\overrightarrow{X_2}$

$r_1 = \left\|\overrightarrow{X_2} - C_1\right\| = \|(0,1) - (0,0)\| = \sqrt{(0-0)^2 + (1-0)^2} = 1$

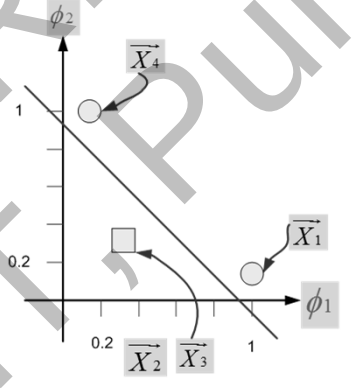
$r_2 = \left\|\overrightarrow{X_2} - C_2\right\| = \|(0,1) - (1,1)\| = \sqrt{(0-1)^2 + (1-1)^2} = 1$

$\phi_1 = \exp(-r_1^2) = \exp(-(1)^2) = 0.3679$

$\phi_2 = \exp(-r_2^2) = \exp(-(1)^2) = 0.3679$

x_1	x_2	r_1	r_2	$\emptyset_1$	$\emptyset_2$
0	0	0	1.4142	1	0.1353
0	1	1	1	0.3679	0.3679
1	0	1	1	0.3679	0.3679
1	1	1.4142	0	0.1353	1

Mapped points



$$\phi = \begin{bmatrix} 1 & 0.1353 & 1 \\ 0.3679 & 0.3679 & 1 \\ 0.3679 & 0.3679 & 1 \\ 0.1353 & 1 & 1 \end{bmatrix}, W = \begin{bmatrix} w_1 \\ w_2 \\ w_0 \end{bmatrix}, D = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Solve this discrimination function to complete the solution:

$\phi \cdot W = D$

$\therefore W = \phi^+ \cdot D = \left[\left(\phi^T \cdot \phi\right)^{-1} \cdot \phi^T\right] \cdot D$

yields:

$$W = \begin{bmatrix} -2.5031 \\ -2.5031 \\ 2.8418 \end{bmatrix}$$

Checking the solution:

$$\begin{bmatrix} 1 & 0.1353 & 1 \\ 0.3679 & 0.3679 & 1 \\ 0.3679 & 0.3679 & 1 \\ 0.1353 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} -2.5031 \\ -2.5031 \\ 2.8418 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Fixing the radius σ

This is usually done using the P-nearest neighbor algorithm. A number P is chosen, and for each center, the P nearest centers are found. The root mean squared distance between the current cluster center and its P nearest neighbors is calculated, and this is the value chosen for σ. So, if the current cluster center is c_j , the value is:

$$\sigma_j = \sqrt{\frac{1}{P}\sum_{i=1}^{P}\left(c_k - c_i\right)^2}$$

A typical value for P is 2, in which case σ is set to be the average distance from the two nearest neighboring cluster centers.

The variable σ defines the width or radius of the bell shape and is something that has to be determined empirically. When the distance from the center of the Gaussian reaches σ, the output drops from 1 to 0.6.

RBFN Algorithm

Step 1	Get n, k ,N,the feature vectors and their target vector, input the number of iterations I, set $i = 0$, set m centers of RBF's as the N exemplar vectors, $msegoal$ and learning rate α	
Step 2	**UNSUPERVISED LEARNING**	
	Using clustering algorithm like k-means clustering find the m cluster centers. Find minimum Euclidean distance of the cluster centers to fix σ (radius)	
Step 3	Compute the ϕ-matrix	
Step 4	**SUPERVISED LEARNING**	
	Choose weights and biases W at random between -0.5 and 0.5	
	Step 5	Compute y_k and error E and mse
	Step 6	Using Delta Rule update all parameters W_{mk} for all m and k at the current iteration
	Step 7	Compute y_k and new value of error E
	Step 8	If $mse \leq msegoal$ stop else continue;
	Step 9	Increment iteration i, if $i < I$ then go to Step 5, else stop

k-means clustering

- The k-means algorithm partitions the data into k clusters. A popular criterion function associated with the k-means algorithm is the sum of squared error. Let k be the number of clusters and n the number of data in the sample $X_1, X_2, \ldots, X_m$. We define the cluster centroid m_i as;

$$m_i = \frac{\sum_{j=1}^{n} \omega_{ij} \cdot \mathrm{x}_j}{\sum_{j=1}^{n} \omega_{ij}}, \forall i$$

with the membership function ω_{ij} indicating whether the data point X_j belongs to a cluster ω_{ij} . The membership values vary according to the type of k-means algorithm. The standard k-means uses an all-or-nothing procedure, that is, ω_{ij}=1, if the data sample X_j belongs to cluster ω_{ij} , else ω_{ij}=0 .

- The membership function must also satisfy the following constraints:

$$\sum_{i=1}^{k} \omega_{ij} = 1, \forall j$$

$$0 < \sum_{j=1}^{n} \omega_{ij} < n, \forall i$$

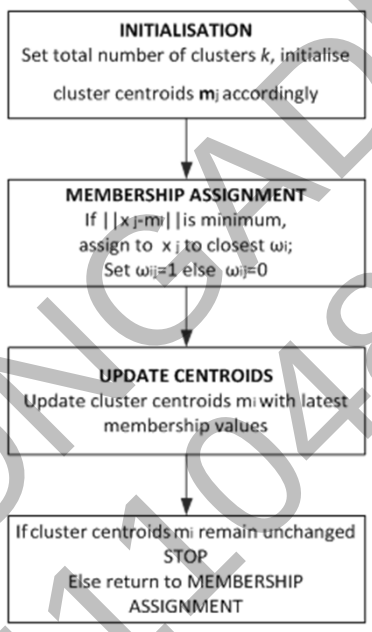
- The k-means algorithm uses a criterion function based on the measure of similarity or distance. For example, using the Euclidean distance that will favor the hyper spherical cluster, a criterion function to minimize is defined by:

$$J = \sum_{i=1}^{k} \sum_{j=1}^{n} \omega_{ij} \cdot \left\| \mathrm{x}_j - m_i \right\|^2$$

which, considering the all-or-nothing membership function, simplifies to:

$$J = \sum_{i=1}^{k} \sum_{\mathrm{x}_j \in \omega_i} \omega_{ij} \cdot \left\| \mathrm{x}_j - m_i \right\|^2$$

- The k-means clustering algorithm is an iterative algorithm that minimizes the criterion function J.
- The initial choice of cluster and measure of similarity or distance affects the way in which the algorithm behaves.
- This type algorithm tends to converge to local minima close to cluster centroids set initially.
- This reinforces the importance of initial consideration with regard to the choice of cluster, keeping in mind that such an algorithm does not converge to the global minimum.

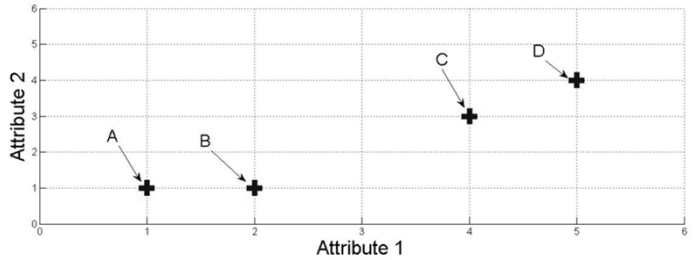


Numerical example of k-means clustering

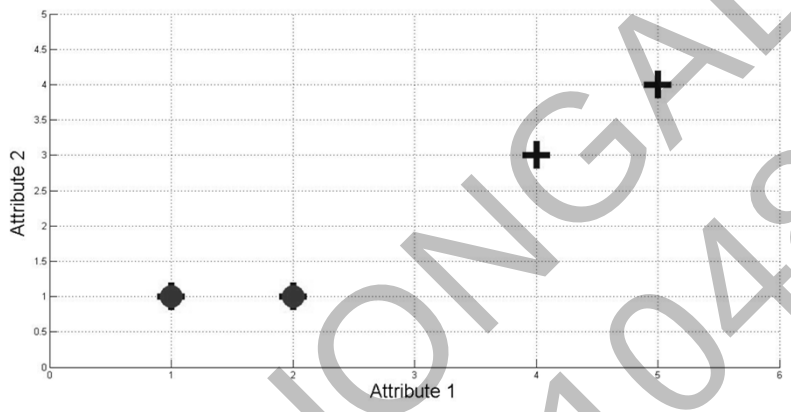
- Suppose we have several objects (4 types of medicines) and each object have two attributes or features as shown in table below.
- Our goal is to group these objects into K=2 group of medicine based on the two features (pH and weight index).

Objects	Attribute 1 (weight index)	Attribute 2 (pH)
Medicine A	1	1
Medicine B	2	1
Medicine C	4	3
Medicine D	5	4

- Each medicine represents one point with two attributes that we can represent as coordinates in an attribute space as shown in the figure below.



1. Initial value of centroids : Suppose we use medicine A and medicine B as the first centroids. Let C1 and C2 denote the coordinate of the centroids, then C1=[1,1] and C2 =[2,1]



2. Objects-Centroids distance : we calculate the distance between cluster centroid to each object. Let us use Euclidean distance, then we have distance matrix at iteration 0 as:

$$D^0 = \begin{bmatrix} 0 & 1 & 3.61 & 5 \\ 1 & 0 & 2.83 & 4.24 \end{bmatrix}, \quad O = \begin{bmatrix} 1 & 2 & 4 & 5 \\ 1 & 1 & 3 & 4 \end{bmatrix} \begin{matrix} Attribute1 \\ Attribute2 \end{matrix}$$

$$C_1 = \begin{bmatrix} 1 & 1 \end{bmatrix}$$

$$C_2 = \begin{bmatrix} 2 & 1 \end{bmatrix}$$

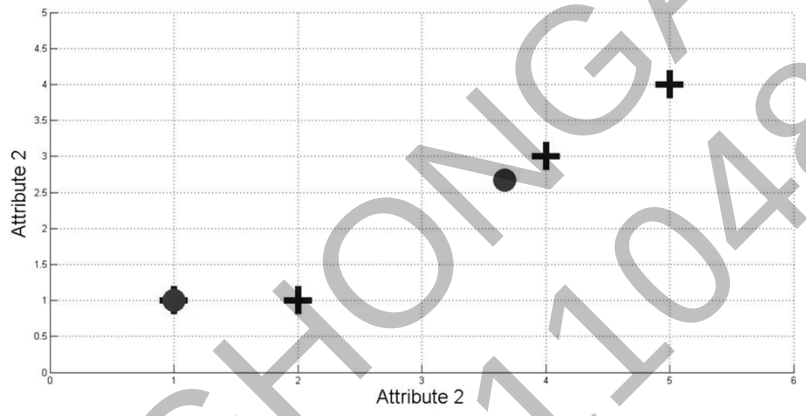
Each column in the distance matrix symbolizes the object. The first row of the distance matrix corresponds to the distance of each object to the first centroid and the second row is the distance of each object to the second centroid. For example, distance from medicine C = (4, 3) to the first centroid C1=(1,1) is $\sqrt{(4-1)^2+(3-1)^2} = 3.61$ and distance with C2=(2,1) is $\sqrt{(4-2)^2+(3-1)^2} = 2.83$ and so on.

3. Objects clustering : We assign each object based on the minimum distance. Thus, medicine A is assigned to group 1, medicine B to group 2, medicine C to group 2 and medicine D to group 2. The element of Group matrix below is 1 if and only if the object is assigned to that group.

$$G^0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \begin{matrix} Group1 \\ Group2 \end{matrix}$$

4. Iteration-1, determine centroids : Knowing the members of each group, now we compute the new centroid of each group based on these new memberships. Group 1 only has one member thus the centroid remains in C1=(1,1). Group 2 now has three members, thus the centroid is the average coordinate among the three members:

$$C_2 = \left(\frac{2+4+5}{3}, \frac{1+3+4}{3}\right) = (3.67, 2.67)$$



5. Iteration-1, Objects-Centroids distances : The next step is to compute the distance of all objects to the new centroids. Similar to step 2, we have distance matrix at iteration 1 as:

$$D^1 = \begin{bmatrix} 0 & 1 & 3.61 & 5 \\ 3.14 & 2.36 & 0.47 & 1.89 \end{bmatrix},$$

$$C_1 = \begin{bmatrix} 1 & 1 \end{bmatrix}$$

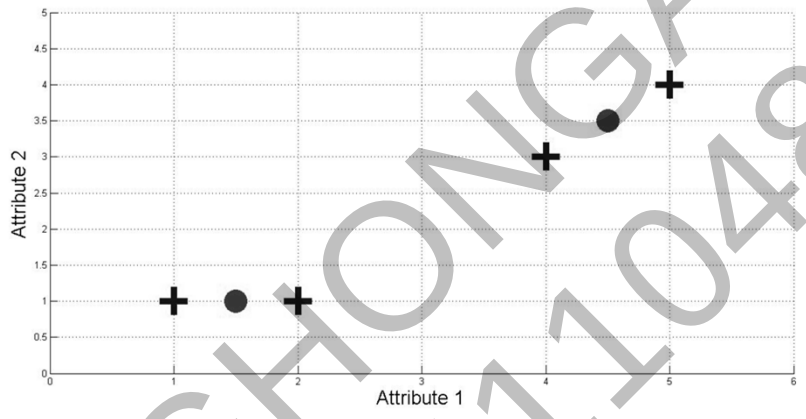
$$C_2 = \begin{bmatrix} 3.67 & 2.67 \end{bmatrix}$$

6. Iteration-1, Objects clustering: Similar to step 3, we assign each object based on the minimum distance. Based on the new distance matrix, we move the medicine B to Group 1 while all the other objects remain unchanged. The Group matrix is shown below:

$$G^1 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{matrix} Group1 \\ Group2 \end{matrix}$$

7. Iteration 2, determine centroids: Now we repeat step 4 to calculate the new centroids coordinate based on the clustering of previous iteration. Group1 and group 2 both has two members, thus the new centroids are

$$C_1 = \left(\frac{1+2}{2}, \frac{1+1}{2}\right) = (1.5, 1) \qquad C_2 = \left(\frac{4+5}{2}, \frac{3+4}{2}\right) = (4.5, 3.5)$$



8. Iteration-2, Objects-Centroids distances : Repeat step 2 again, we have new distance matrix at iteration 2 as:

$$D^2 = \begin{bmatrix} 0.5 & 0.5 & 3.2 & 4.61 \\ 4.3 & 3.54 & 0.71 & 0.71 \end{bmatrix},$$

$$C_1 = [1.5 \quad 1]$$

$$C_2 = [4.5 \quad 3.5]$$

9. Iteration-2, Objects clustering: Again, we assign each object based on the minimum distance:

$$G^2 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{matrix} Group1 \\ Group2 \end{matrix}$$

We see that hence the objects do not move anymore and the algorithm has reached a stable point.

RBFN MATLAB DEMO

If only life were so simple…

- How do we choose number of clusters ? Similar to problem of selecting number of hidden nodes for MLP
- What type of pre-processing is best?
- Does the clustering method work for the data? E.g might be better to fix σ and try again.

There is NO general answer: each choice will be problem-specific. The only info you have is your performance measure.

Comparison

MLP

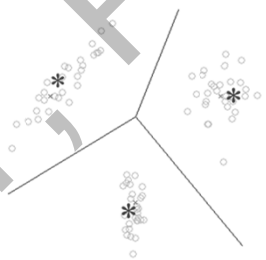
- Feedforward network
- Universal approximator
- Can have more than one hidden layers
- Completely supervised learning
- Network dimension according to dimension of data
- All neurons output and hidden layer , share common neuronal model
- MLPs construct global approximations to non-linear input output mapping

RBFN

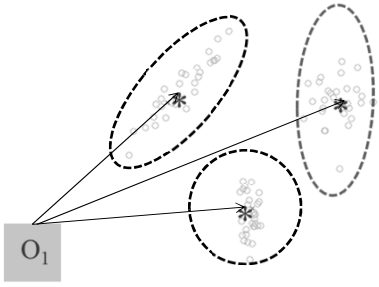
- Feedforward network
- Universal approximator
- Only one hidden layer
- Learning in two phases: unsupervised learning (clustering) followed by supervised learning (delta rule)
- Network dimension according to information content
- Hidden layer neurons are completely different than output neurons
- RBF using exponentially decaying localized non-linearities (Gaussian) construct local approximations to input-output mapping

Comparison

MLP



RBFN



Applications of RBFN

1. **Regression**
2. **Pattern Classification**

Thank You!