

NEURAL NETWORKS (CS010 805G02)

Radial Basis Functions – Mod 3

Shiney Thomas
AP,CSE,AJCE

RBF NETWORK

- This is becoming an increasingly popular neural network with diverse applications and is probably the main rival to the multi-layered perceptron
- Much of the inspiration for RBF networks has come from traditional statistical pattern classification techniques



RADIAL FUNCTION (FROM WIKIPEDIA)

- In mathematics, a **radial function** is a function defined on a Euclidean space \mathbf{R}^n whose value at each point depends only on the distance between that point and the origin. For example, a radial function Φ in two dimensions has the form

$$\Phi(x,y) = \varphi(r), \quad r = \sqrt{x^2 + y^2}$$

where φ is a function of a single non-negative real variable.

RADIAL BASIS FUNCTION (FROM WIKIPEDIA)

- A **radial basis function (RBF)** is a real-valued function whose value depends only on the distance from the origin, so that $\varphi(x) = \varphi(\|x\|)$ or alternatively on the distance from some other point c , called a *center*, so that

$$\varphi(x, c) = \varphi(\|x - c\|)$$

- Any function φ that satisfies the property $\varphi(x) = \varphi(\|x\|)$ is a radial function.
- The norm is usually Euclidean distance, although other distance functions are also possible.

RBF NETWORK

- The basic architecture for a RBF is a 3-layer network, as shown in Fig.
- The input layer is simply a fan-out layer and does no processing.
- The second or hidden layer performs a non-linear mapping from the input space into a (usually) higher dimensional space in which the patterns become linearly separable.
- The final layer performs a simple weighted sum with a linear output.



RADIAL BASIS FUNCTION NETWORK

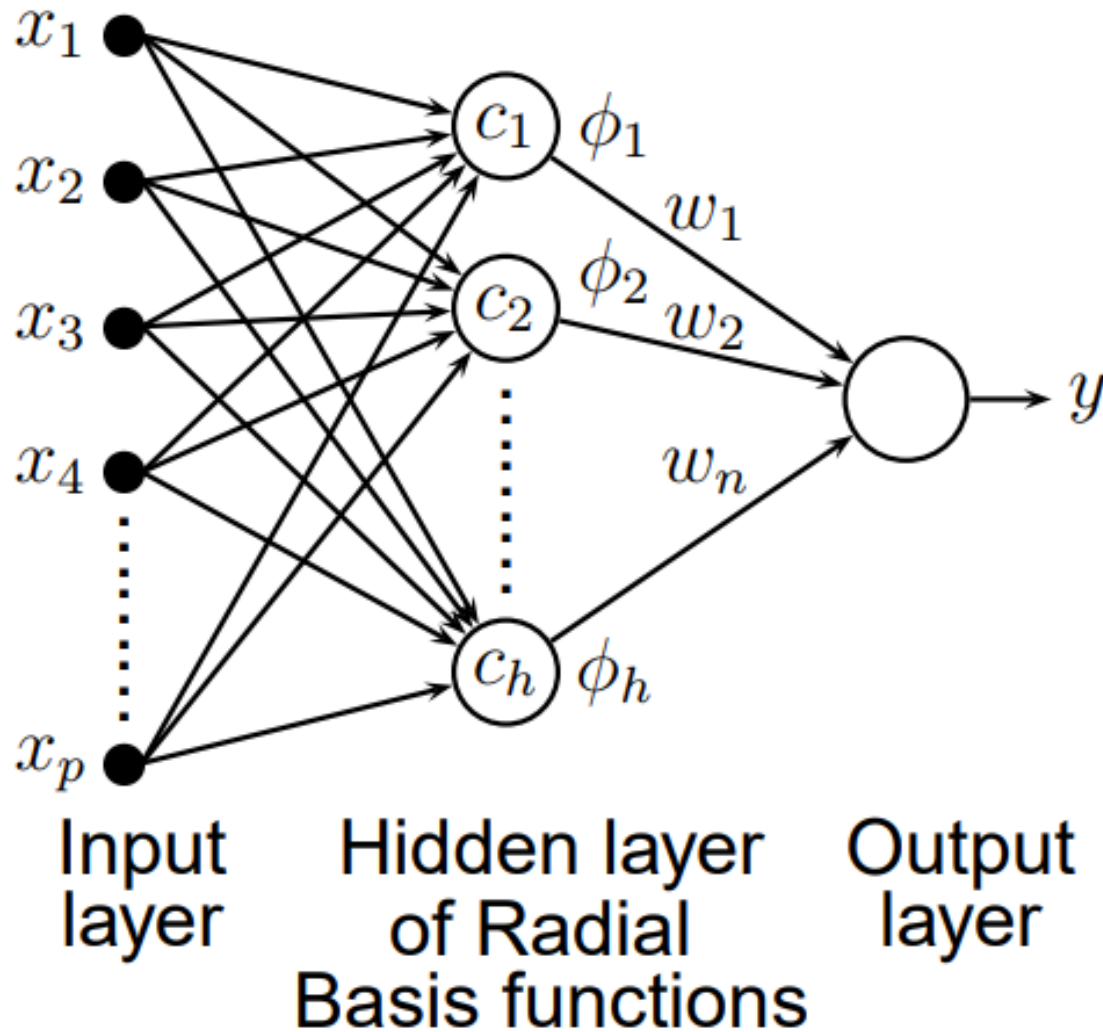
- Approximate function with linear combination of Radial basis functions

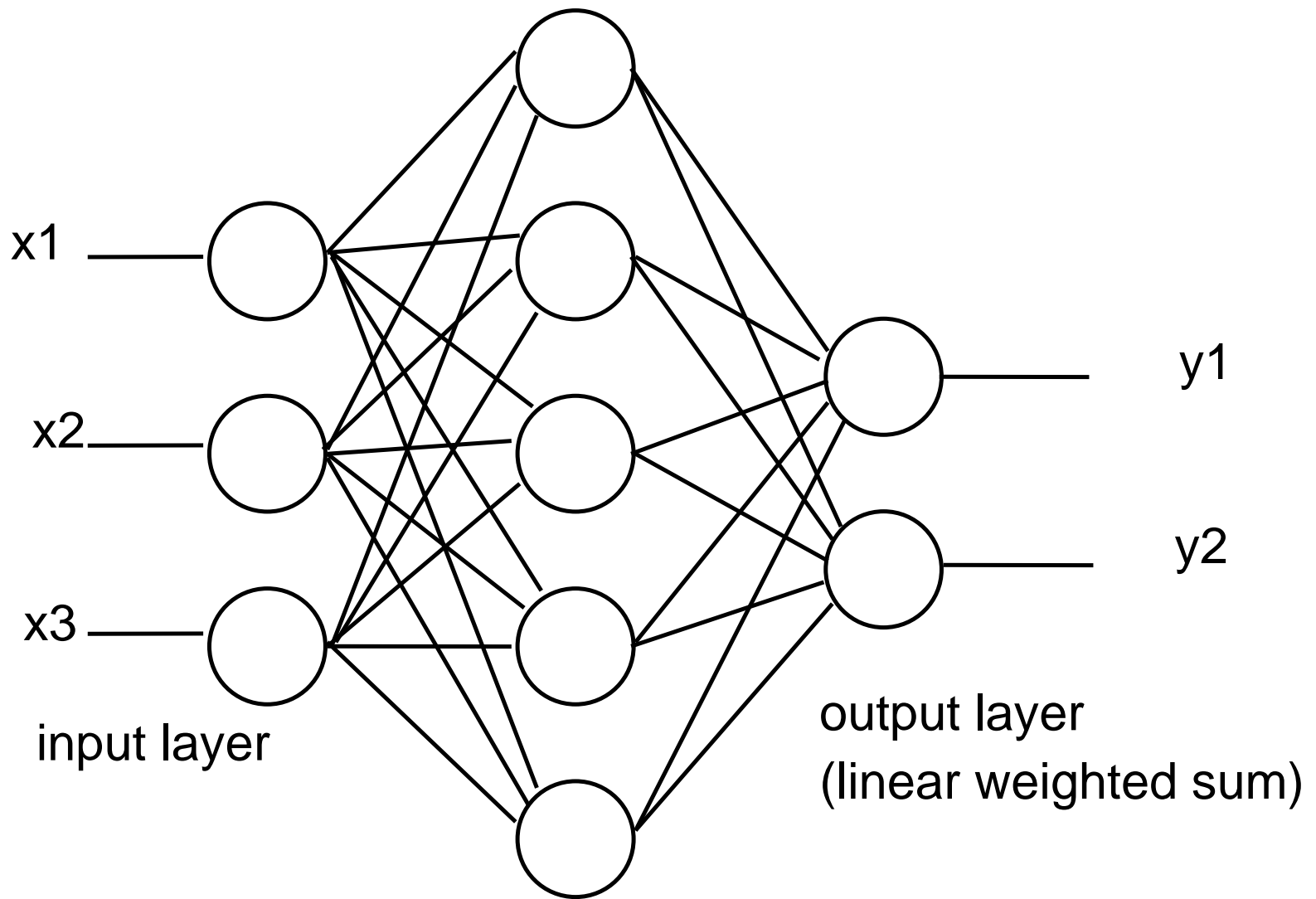
$$F(x) = \sum w_i h(x)$$

- $h(x)$ is mostly Gaussian function



NETWORK ARCHITECTURE





hidden layer

(weights correspond to cluster centre,
output function usually Gaussian)



COVERS THEOREM

“A complex pattern-classification problem cast in high-dimensional space nonlinearly is more likely to be linearly separable than in a low dimensional space”

(Cover, 1965).



INTRODUCTION TO COVER'S THEOREM

- Let X denote a set of N patterns (points)
 $x_1, x_2, x_3, \dots, x_N$
- Each point is assigned to one of two classes:
 X^+ and X^-
- This dichotomy(binary partition) is **separable** if there exist a surface that separates these two classes of points.



INTRODUCTION TO COVER'S THEOREM – CONT'D

- For each pattern $x \in X$ define the next vector:
$$\varphi(x) = [\varphi_1(x), \varphi_2(x), \dots, \varphi_M(x)]^T$$
- The vector $\varphi(x)$ maps points in a p -dimensional input space into corresponding points in a new space of dimension m .
- Each $\varphi_i(x)$ is a hidden function, i.e., a **hidden unit**



INTRODUCTION TO COVER'S THEOREM

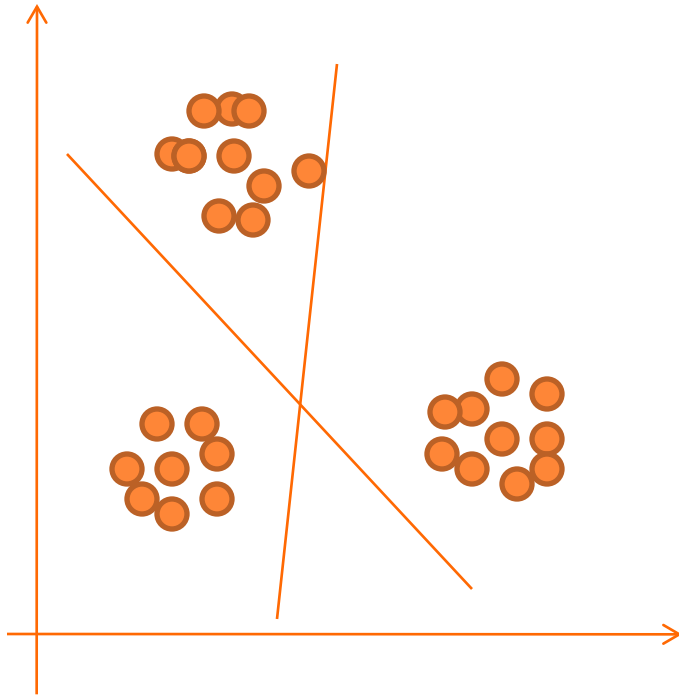
– CONT'D

- A dichotomy $\{X^+, X^-\}$ is said to be **ϕ -separable** if there exist a m -dimensional vector w such that we may write (Cover, 1965):
 - $w^T \phi(x) \geq 0, x \in X^+$
 - $w^T \phi(x) < 0, x \in X^-$
 - The hyperplane defined by $w^T \phi(x) = 0$, is the separating surface between the two classes.

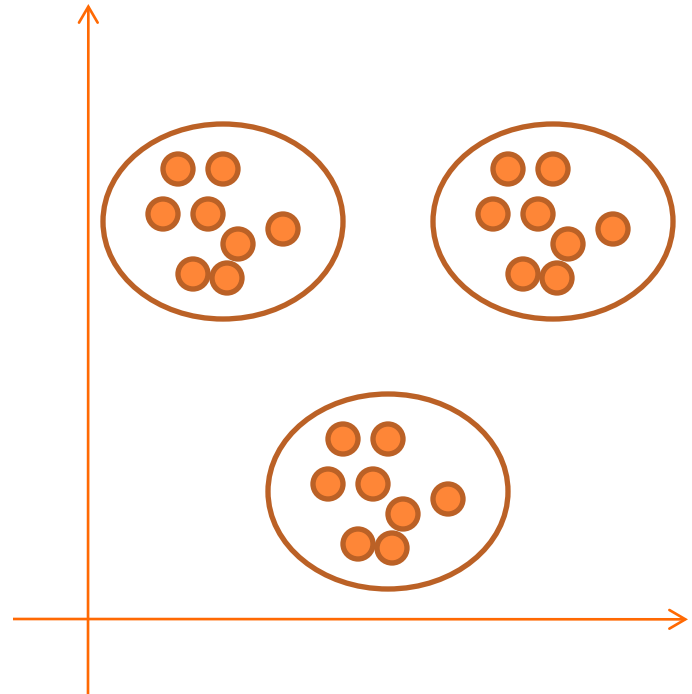


RBF NETWORKS FOR CLASSIFICATION

○ MLP



○ RBF



RBF NETWORKS FOR CLASSIFICATION

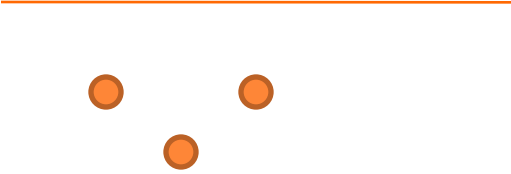
CONTD...

- An MLP naturally separates the classes with hyperplanes in the Input space
- RBF would be to separate class distributions by localizing radial basis functions
- Types of separating surfaces are
 - Hyperplane-linearly separable
 - Spherically separable-Hypersphere
 - Quadratically separable-Quadrics



Hyperplane-linearly
separable

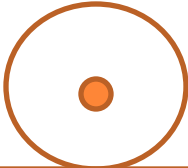
X X



A horizontal line separates two regions. Above the line are two 'X' characters. Below the line are three orange dots arranged in a triangular pattern.

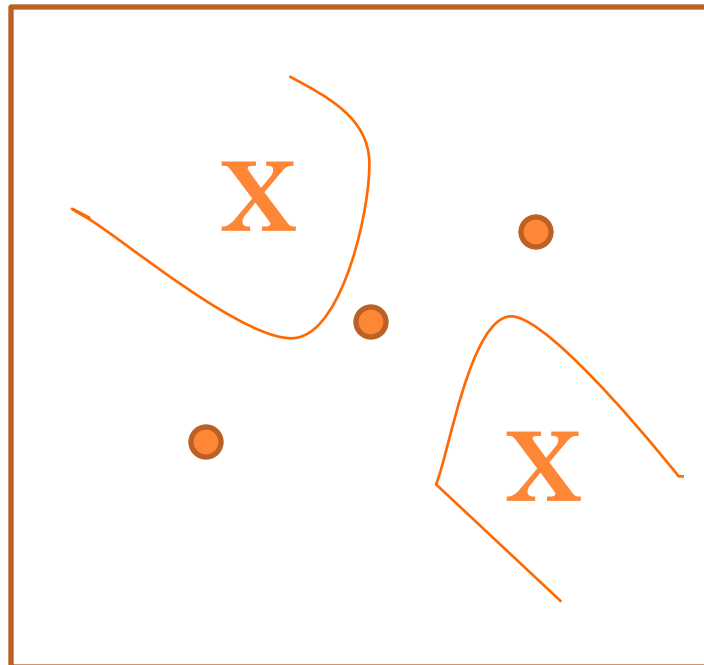
Hypersphere-
spherically separable

X X
X X



Four 'X' characters are arranged in a square. In the center of the square is an orange dot surrounded by a circle.

Quadratically separable-
Quadrics



WHAT HAPPENS IN HIDDEN LAYER?

- The patterns in the input space form clusters
- If the centers of these clusters are known then the distance from the cluster center can be measured
- The most commonly used radial basis function is a Gaussian function
- In a RBF network r is the distance from the cluster centre



CLUSTERING

- The unique feature of the RBF network is the process performed in the hidden layer.
- The idea is that the patterns in the input space form clusters.
- If the centres of these clusters are known, then the distance from the cluster centre can be measured.
- Furthermore, this distance measure is made non-linear, so that if a pattern is in an area that is close to a cluster centre it gives a value close to 1.



CLUSTERING

- Beyond this area, the value drops dramatically.
- The notion is that this area is radially symmetrical around the cluster centre, so that the non-linear function becomes known as the radial-basis function.



COMMONLY USED RADIAL BASIS FUNCTIONS

1. Gaussian Functions:

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

width parameter $\sigma > 0$

2. Multi-Quadric Functions:

$$\phi(r) = (r^2 + \sigma^2)^{1/2}$$

parameter $\sigma > 0$

3. Generalized Multi-Quadric Functions:

$$\phi(r) = (r^2 + \sigma^2)^\beta$$

parameters $\sigma > 0, 1 > \beta > 0$

COMMONLY USED RADIAL BASIS FUNCTIONS

4. Inverse Multi-Quadric Functions:

$$\phi(r) = (r^2 + \sigma^2)^{-1/2} \quad \text{parameter } \sigma > 0$$

5. Generalized Inverse Multi-Quadric Functions:

$$\phi(r) = (r^2 + \sigma^2)^{-\alpha} \quad \text{parameters } \sigma > 0, \alpha > 0$$

6. Thin Plate Spline Function:

$$\phi(r) = r^2 \ln(r)$$

7. Cubic Function:

$$\phi(r) = r^3$$

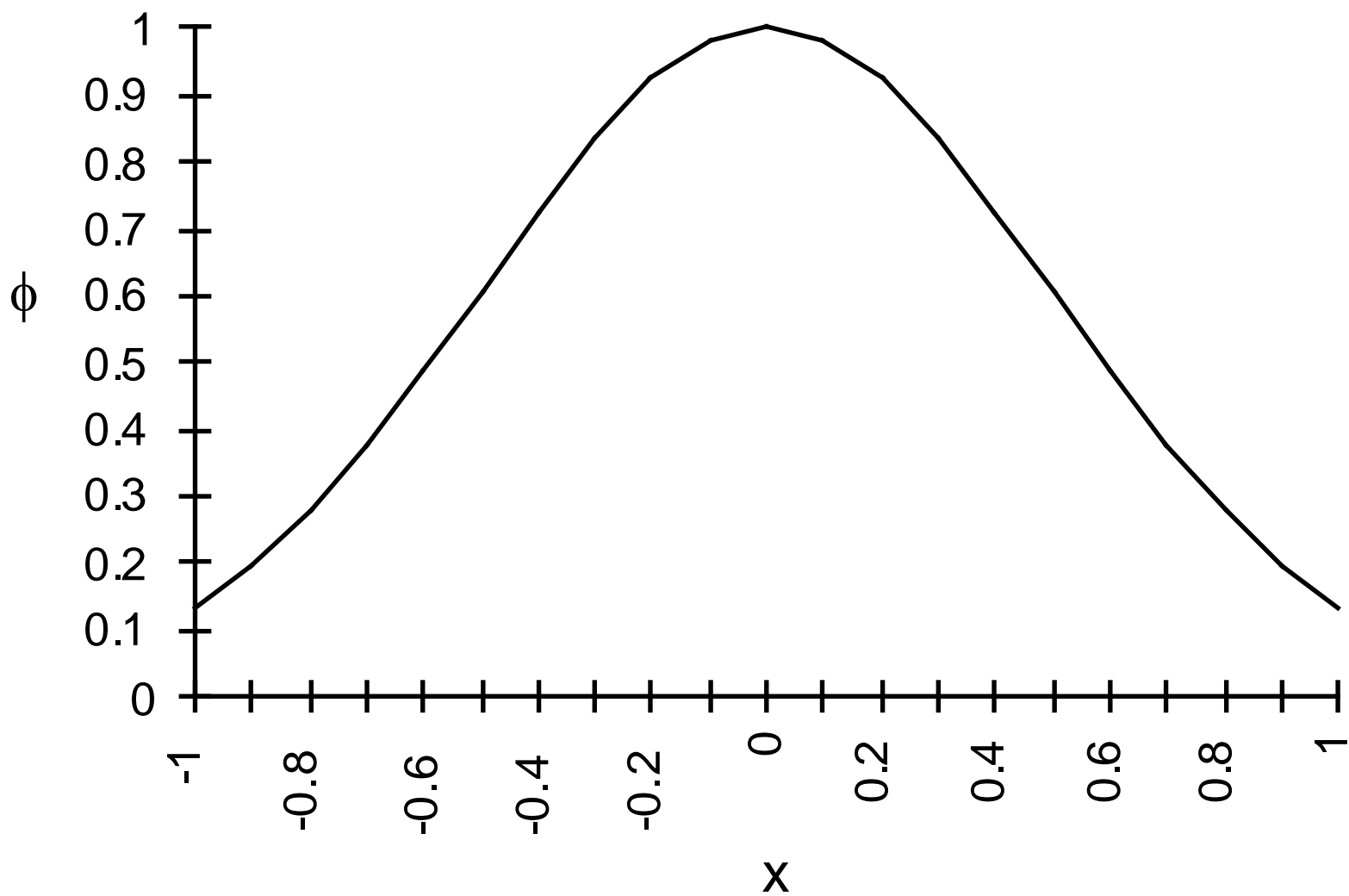
8. Linear Function:

$$\phi(r) = r$$

GAUSSIAN FUNCTION

- The most commonly used radial-basis function is a Gaussian function
- In a RBF network, r is the distance from the cluster centre.
- The equation represents a Gaussian bell-shaped curve, as shown in Fig.





DISTANCE MEASURE

- The distance measured from the cluster centre is usually the Euclidean distance.
- For each neuron in the hidden layer, the weights represent the co-ordinates of the centre of the cluster.
- Therefore, when that neuron receives an input pattern, X , the distance is found using the following equation:

$$r_j = \sqrt{\sum_{i=1}^n (x_i - w_{ij})^2}$$



WIDTH OF HIDDEN UNIT BASIS FUNCTION

$$(hidden_unit)\varphi_j = \exp\left(-\frac{\sum_{i=1}^n (x_i - w_{ij})^2}{2\sigma^2}\right)$$

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



TRAINING THE OUTPUT LAYER

- The output layer is trained using the least mean square algorithm, which is a gradient descent technique
- Given input signal vector $x(n)$ and desired response $d(n)$
- Set initial weights $w(x)=0$
- For $n=1,2,\dots$
 Compute
 $e(n)=\text{error}=d - w^t x$
 $w(n+1)=w(n)+c.x(n).e(n)$



SIMILARITIES BETWEEN RBF AND MLP

- Both are feedforward
- Both are universal approximators
- Both are used in similar application areas

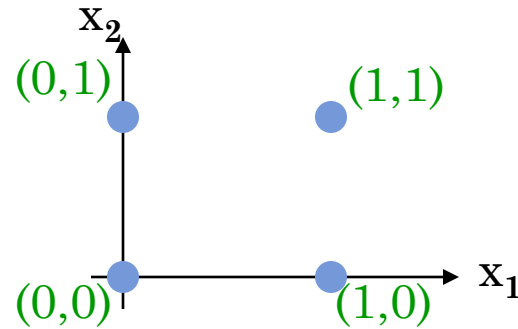


DIFFERENCES BETWEEN MLP AND RBF

MLP	RBF
Can have any number of hidden layer	Can have only one hidden layer
Can be fully or partially connected	Has to be mandatorily completely connected
Processing nodes in different layers shares a common neural model	Hidden nodes operate very differently and have a different purpose
Argument of hidden function activation function is the inner product of the inputs and the weights	The argument of each hidden unit activation function is the Euclidean norm distance between the input and the center of that unit
Construct Global approx to non linear input-output mapping	RBF networks using exponentially decaying localized nonlinearities construct local approx to input-output mapping
Training is slower compared to RBF	Training is comparatively faster than MLP
After training MLP is much faster than RBF	After training RBF is much slower than MLP

EXAMPLE: THE XOR PROBLEM

- Input space:



- Output space:



- Construct an RBF pattern classifier such that:
 $(0,0)$ and $(1,1)$ are mapped to 0, class C1
 $(1,0)$ and $(0,1)$ are mapped to 1, class C2



Example: the XOR problem

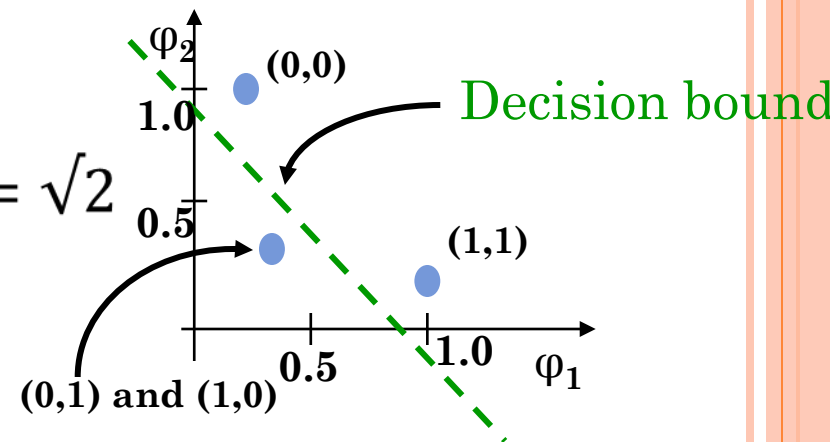
- In the feature (hidden layer) space:

no.of patterns=4

$M=2$

$\mu_1=(0,0); \mu_2=(1,1)$

$$d_{\max} = \sqrt{((0-1)^2 + (0-1)^2)} = \sqrt{2}$$



- When mapped into the feature space $\langle \phi_1, \phi_2 \rangle$ (hidden layer), C1 and C2 become *linearly separable*. So a linear classifier with $\phi_1(\mathbf{x})$ and $\phi_2(\mathbf{x})$ as inputs can be used to solve the XOR problem.

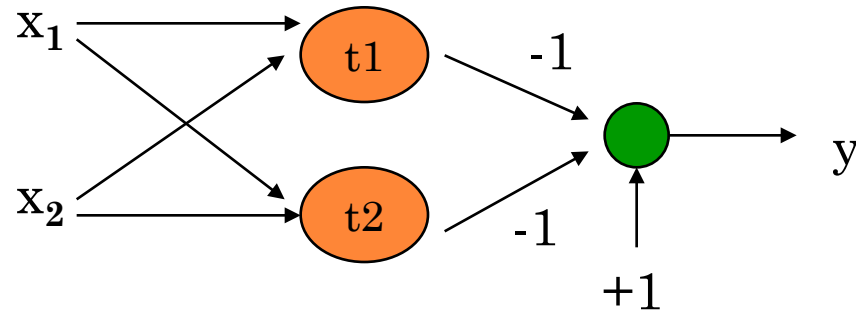
RBF NN for the XOR problem

$$\varphi_1(x) = e^{-\|x - \mu_1\|^2}$$

$$\varphi_2(x) = e^{-\|x - \mu_2\|^2}$$

with $\mu_1 = (0,0)$ and $\mu_2 = (1,1)$

The centres for the two hidden units are set at $c_1 = 0,0$ and $c_2 = 1,1$, and the value of radius σ is chosen such that $2\sigma^2 = 1$.



Pattern	X1	X2	φ_1	φ_2
1	0	0		
2	0	1		
3	1	0		
4	1	1		



RBF NETWORK PARAMETERS

- **What do we have to learn for a RBF NN with a given architecture?**
 - The centers of the RBF activation functions
 - the spreads of the Gaussian RBF activation functions
 - the weights from the hidden to the output layer
- Different learning algorithms may be used for learning the RBF network parameters. We describe three possible methods for learning centers, spreads and weights.



LEARNING STRATEGIES

1. Fixed centers selected at Random
 2. Self organized selection of centers
 3. Supervised selection of centers
 4. Strict interpolation with Regularization
- } Hybrid learning Process

First three methods pertain to RBF network whose formulation is based on interpolation theory.

The last one combines elements of regularization theory and kernel regression estimation theory

FIXED CENTERS AT RANDOM

- **Centers:** are selected at random
 - **centers** are chosen randomly from the training set
- **Spreads:** are chosen by **normalization**:

$$\sigma = \frac{\text{Maximum distance between any 2 centers}}{\sqrt{\text{number of centers} * 2}} = \frac{d_{\max}}{\sqrt{m_1 * 2}}$$

- Then the activation function of hidden neuron becomes:

i

$$\varphi_i(\|x\|) = \exp\left(-\frac{m_1}{d_{\max}^2} \|x - \mu_i\|^2\right)$$



FIXED CENTERS AT RANDOM

- **Weights:** are computed by means of the **pseudo-inverse method**.
 - For an example (x_i, d_i) consider the output of the network

$$y(x_i) = w_1 \varphi_1(\|x_i - t_1\|) + \dots + w_{m1} \varphi_{m1}(\|x_i - t_{m1}\|)$$

- We would like $y(x_i) = d_i$ for each example, that is

$$w_1 \varphi_1(\|x_i - t_1\|) + \dots + w_{m1} \varphi_{m1}(\|x_i - t_{m1}\|) = d_i$$



FIXED CENTERS AT RANDOM

- This can be re-written in matrix form for one example

$$[\varphi_1(\|x_i - t_1\|) \dots \varphi_{m_1}(\|x_i - t_{m_1}\|)] [w_1 \dots w_{m_1}]^T = d_i$$

and

$$\begin{bmatrix} \varphi_1(\|x_1 - t_1\|) \dots \varphi_{m_1}(\|x_1 - t_{m_1}\|) \\ \dots \\ \varphi_1(\|x_N - t_1\|) \dots \varphi_{m_1}(\|x_N - t_{m_1}\|) \end{bmatrix} [w_1 \dots w_{m_1}]^T = [d_1 \dots d_N]^T$$

for all the examples at the same time



FIXED CENTERS AT RANDOM

let

$$\Phi = \begin{bmatrix} \varphi_1(\|x_1 - t_1\|) & \dots & \varphi_{m1}(\|x_N - t_{m1}\|) \\ & \dots & \\ \varphi_1(\|x_N - t_1\|) & \dots & \varphi_{m1}(\|x_N - t_{m1}\|) \end{bmatrix}$$

then we can write

$$\Phi \begin{bmatrix} w_1 \\ \dots \\ w_{m1} \end{bmatrix} = \begin{bmatrix} d_1 \\ \dots \\ d_N \end{bmatrix}$$

If Φ^+ is the pseudo-inverse of the Φ matrix we obtain the weights using the following formula

$$[w_1 \dots w_{m1}]^T = \Phi^+ [d_1 \dots d_N]^T$$

SELF ORGANIZED SELECTION OF CENTERS

- **k-means clustering algorithm for finding the centers**

- 1 **Initialization**: $t_k(0)$ random $k = 1, \dots, m_1$
- 2 **Sampling**: draw x from input space
- 3 **Similarity matching**: find index of center closer to x

$$k(x) = \arg \min_k \|x(n) - t_k(n)\|_{k=1, \dots, m_1}$$

- 4 **Updating**: adjust centers

$$t_k(n+1) = \begin{cases} t_k(n) + \eta[x(n) - t_k(n)] & \text{if } k = k(x) \\ t_k(n) & \text{otherwise} \end{cases}$$



SELF ORGANIZED SELECTION OF CENTERS

- *k- Means clustering* – a special case of a competitive learning process known as self-organizing map
- Disadvantage – it can achieve only a local optimum solution that depends on the initial choice of centers
 - Enhanced k-means clustering alg.
- *Hybrid Learning Process:*
 - **Clustering** for finding the **centers**.
 - **Spreads** chosen by normalization.
 - **LMS algorithm** for finding the **weights** of the output layer .



SUPERVISED SELECTION OF CENTERS

- RBF centers and all free parameters undergo supervised learning
- Most generalized form
- Process used is error-correction learning, which can be implemented using Gradient descent procedure
- Value of cost fn, $\mathcal{E} = \frac{1}{2} \sum_{j=1}^N e_j^2$
- Where N =size of training sample, e_j is error signal

$$e_j = d_j - F^*(\mathbf{x}_j)$$

$$= d_j - \sum_{i=1}^M w_i G(\|\mathbf{x}_j - \mathbf{t}_i\|_{C_i})$$

APPLICATION: FACE RECOGNITION

- The problem:
 - Face recognition of persons of a known group in an indoor environment.
- The approach:
 - Learn face classes over a wide range of poses using an RBF network.



DATASET

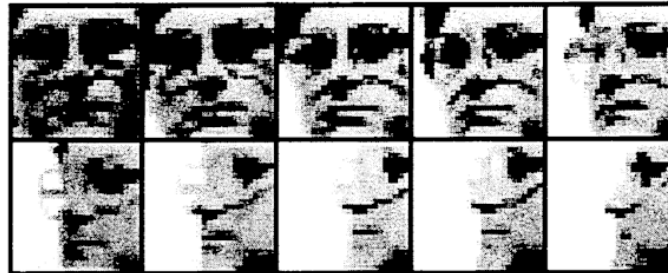
o database

- **100 images of 10 people** (8-bit grayscale, resolution 384 x 287)
- for each individual, 10 images of head in different pose **from face-on to profile**
- Designed to assess performance of **face recognition techniques** when pose variations occur

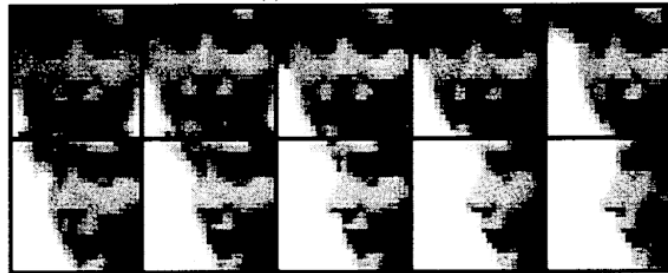


DATASETS

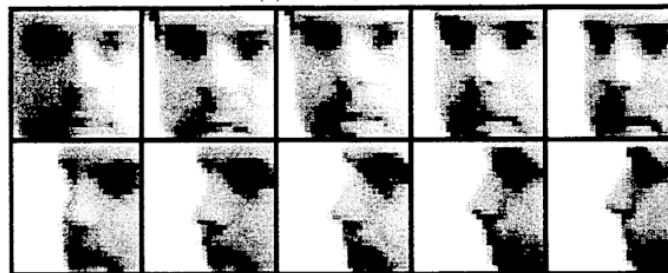
All ten images for
classes 0-3 from the
Sussex database,
nose-centred and
subsampled to 25x25
before preprocessing



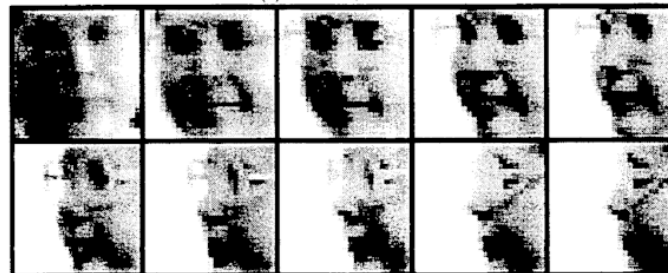
(a) Class 0, 25x25



(b) Class 1, 25x25



(c) Class 2, 25x25



(d) Class 3, 25x25



APPROACH: FACE UNIT RBF

- A face recognition unit RBF neural networks is trained to recognize a single person.
- Training uses examples of images of the person to be recognized as positive evidence, together with selected confusable images of other people as negative evidence.



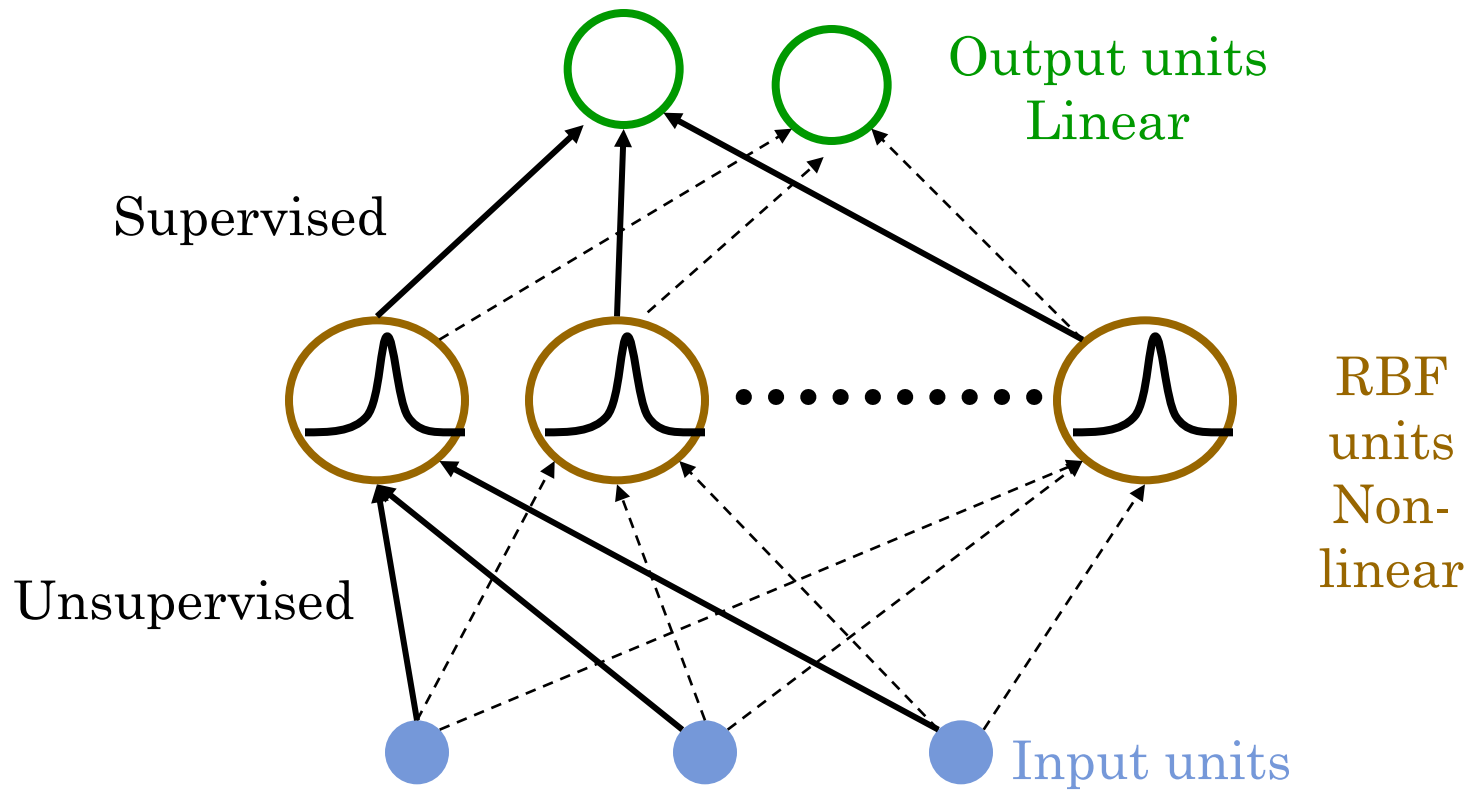
NETWORK ARCHITECTURE

- Input layer contains 25×25 inputs which represent the pixel intensities (normalized) of an image.
- Hidden layer contains $p+a$ neurons:
 - p hidden pro neurons (receptors for positive evidence)
 - a hidden anti neurons (receptors for negative evidence)
- Output layer contains two neurons:
 - One for the particular person.
 - One for all the others.

The output is discarded if the absolute difference of the two output neurons is smaller than a parameter R .



RBF ARCHITECTURE FOR ONE FACE RECOGNITION



HIDDEN LAYER

- Hidden nodes can be:
 - Pro neurons: Evidence for that person.
 - Anti neurons: Negative evidence.
- The number of pro neurons is equal to the positive examples of the training set. For each pro neuron there is either one or two anti neurons.
- Hidden neuron model: Gaussian RBF function.



TRAINING AND TESTING

- **Centers:**
 - of a pro neuron: the corresponding positive example
 - of an anti neuron: the negative example which is most similar to the corresponding pro neuron, with respect to the Euclidean distance.
- **Spread:** average distance of the center from all other centers.
So the spread σ_n of a hidden neuron n is

$$\sigma_n = \frac{1}{H\sqrt{2}} \sum_h \|t^n - t^h\|$$

where H is the number of hidden neurons and t^i is the center of neuron i .

- **Weights:** determined using the pseudo-inverse method.
- A RBF network with 6 pro neurons, 12 anti neurons, and R equal to 0.3, discarded 23 per cent of the images of the test set and classified correctly 96 per cent of the non discarded images.

