# Radial Basis Function Networks

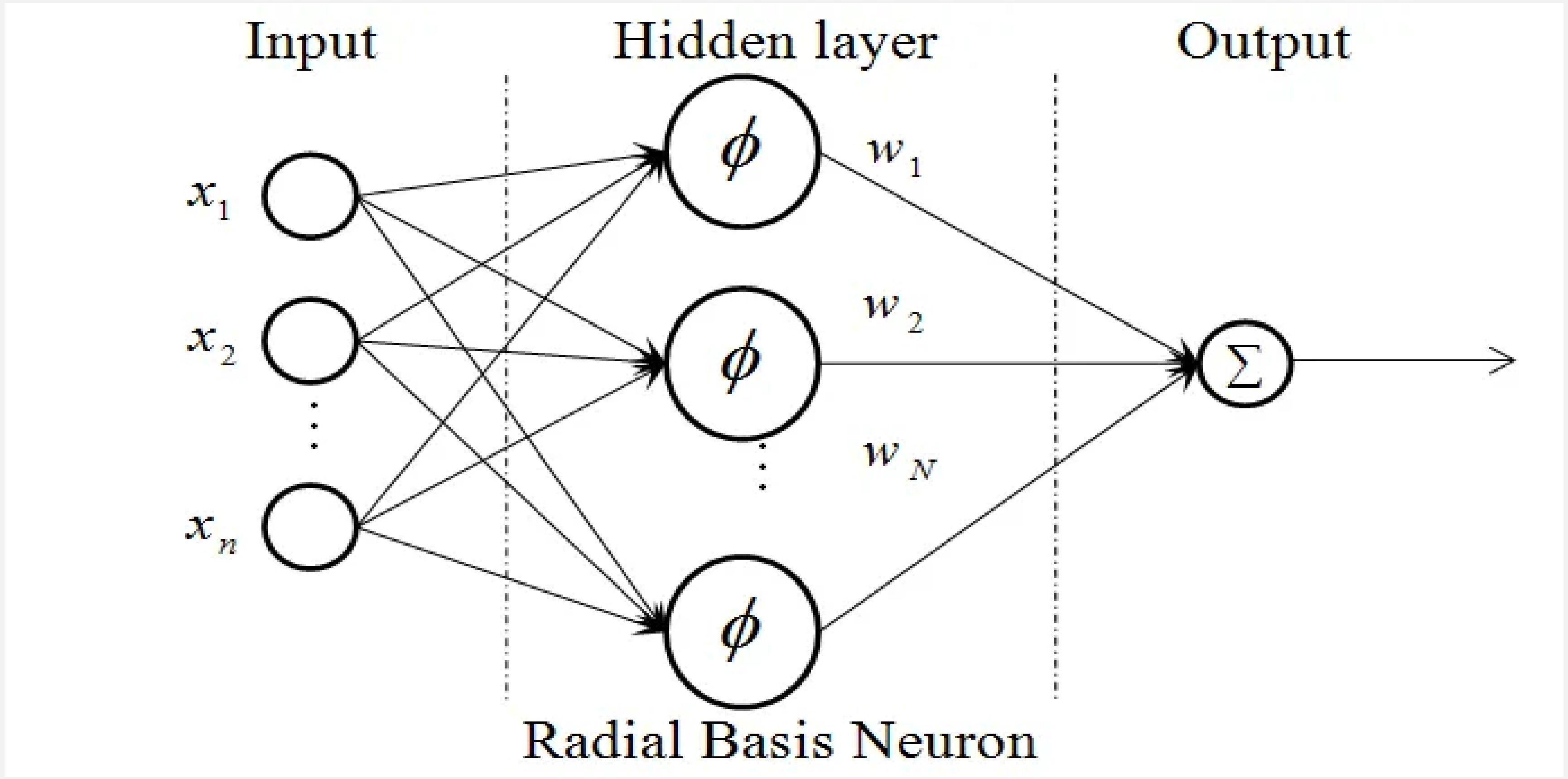
## Introduction

- Radial Basis Function (RBF) Network adapts a completely different approach to design of a neural network for function approximation or classification.
- In RBF network, the hidden neurons provide a set of “functions” that constitute an arbitrary “basis” for the input feature vectors when they are expanded into hidden space. These functions are called radial-basis functions.
- The construction of RBF network involves three layers, the second layer is only one hidden layer along with input and output layer in the network.
- Hidden layer applies a nonlinear transformation from the input space to the hidden space in order to make data linearly separable in a higher dimension.
- The output layer applies supervised learning for classification of linearly separable data like perceptron.

## The main concept of RBFN

- Cover's Theorem on the Separability of Patterns: In a 1965 paper, Cover proposed that nonlinearly separable data can be transformed into a higher-dimensional space by hidden layer neurons using radial basis functions, making the data linearly separable and easier to classify in the output layer, similar to the Perceptron model.
- Another important fact that the dimension of the hidden space is directly related to the capacity of the network to approximate a smooth input-output mapping.
- The higher the dimension of the hidden space, the more accurate the approximation will be.

# Radial Basis Function (RBF) Network



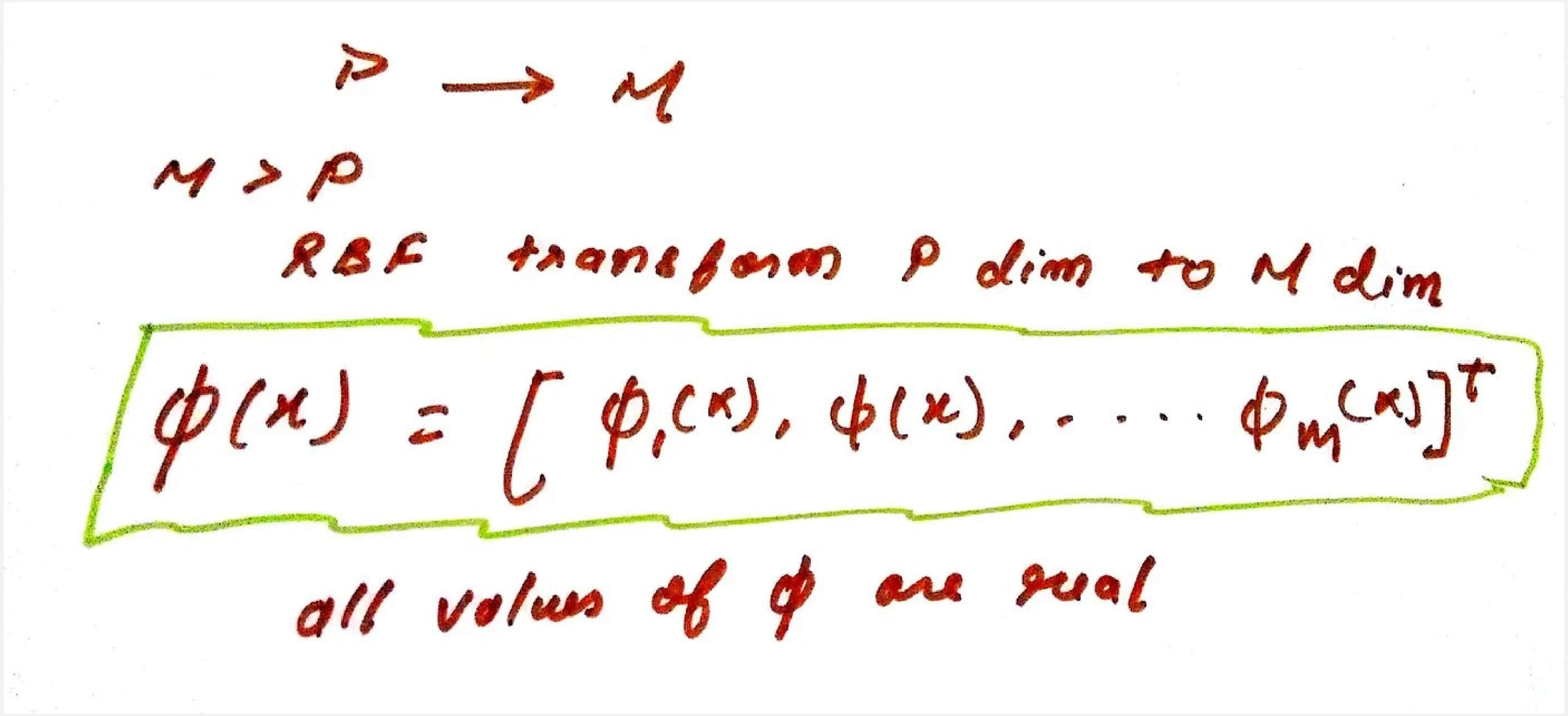
## Principle of RBFN

- Always consists of only one hidden layer.
- The hidden layer of RBF is nonlinear.
- The computation nodes in the hidden and output layer of an RBF are quite different.
- RBF performs nonlinear transformation over input vector before they are fed for classification with help of radial basis functions transformations in the hidden layer.
- Increases Dimensionality of feature vector in the hidden layer.

## Continue…

- The activation function of the hidden layer computes the Euclidean norm between the input vector and the center unit.
- RBF uses exponentially decaying nonlinearities that construct a local approximation to nonlinear Input/Output mappings.
- In general there is always a possibility that the input feature vector which is not linearly separable on current dimensions can become linearly separable while projecting on higher dimensions.

P: Input feature vector dimension and
M: Hidden layer number of neurons dimension



$P \longrightarrow M$

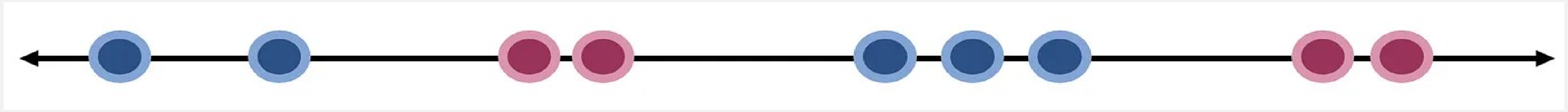
$M > P$

RBF transform P dim to M dim

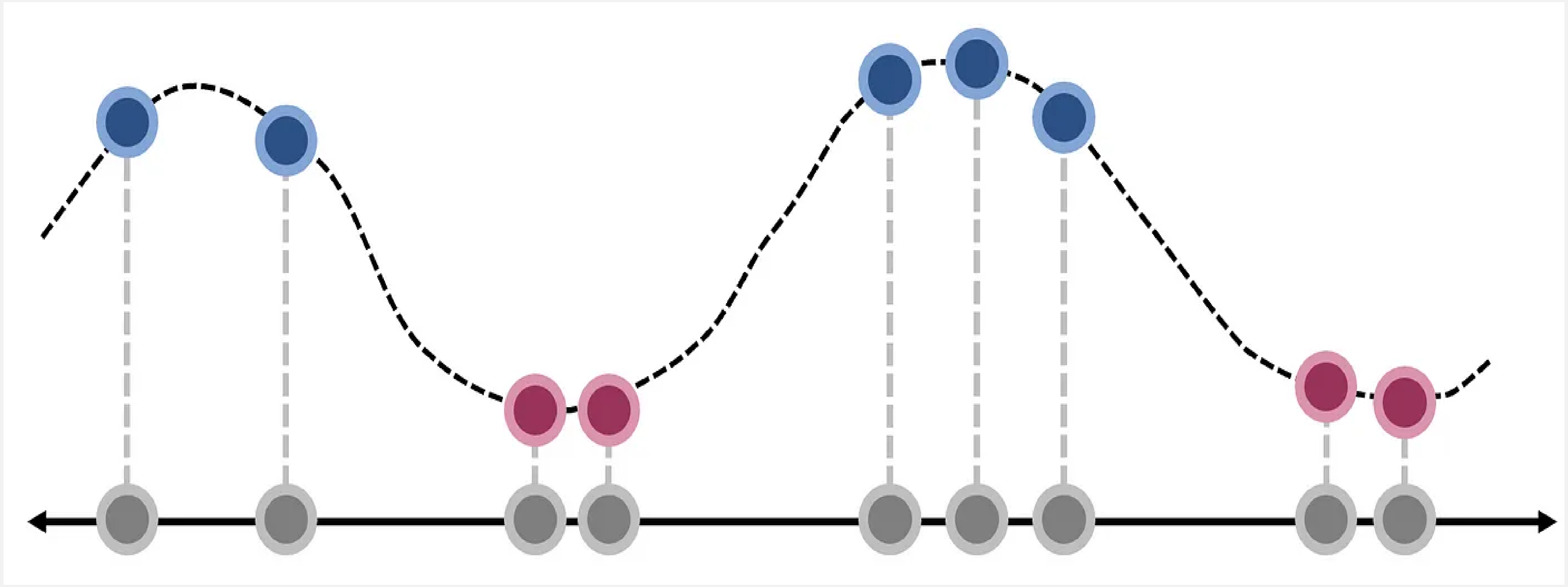
$$\boxed{\phi(x) = [\phi_1(x), \phi(x), \ldots \phi_M(x)]^T}$$

all values of $\phi$ are real

- Here is a set of one-dimensional data: your task is to find a way to perfectly separate the data into two classes with one line.

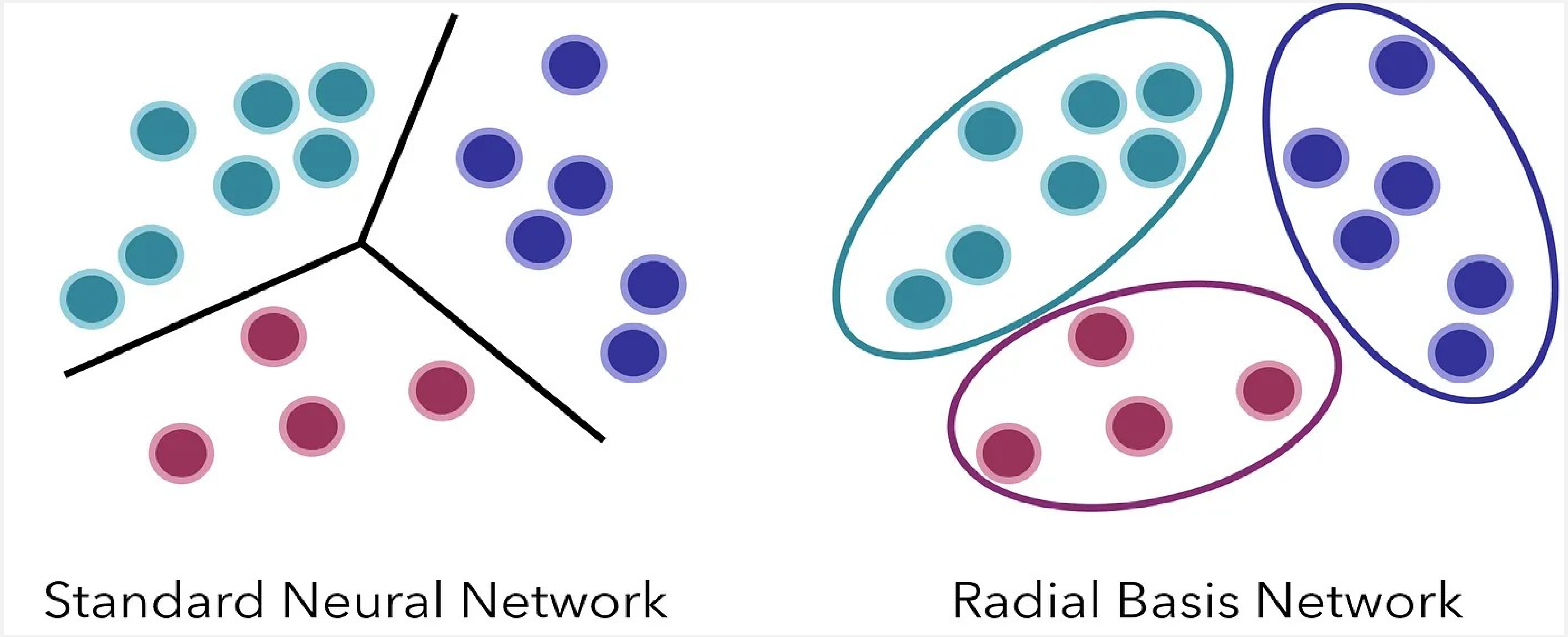


- At first glance, this may appear to be an impossible task, but it is only so if we restrict ourselves to one dimension.
- Let’s introduce a sinusoidal function $\varphi(x)$ and map each value of x to its corresponding output.
- Conveniently, this makes all the blue points higher and the red points lower at just the right locations.
- We can then draw a decision line that clearly divides the classes into two parts.



- This solution seems very difficult, but we can actually generalize it with the help of radial basis functions (RBFs).

- An RBF inherently is simply a function whose points are defined as distances from a center.


Standard Neural Network
Radial Basis Network

## RBFN

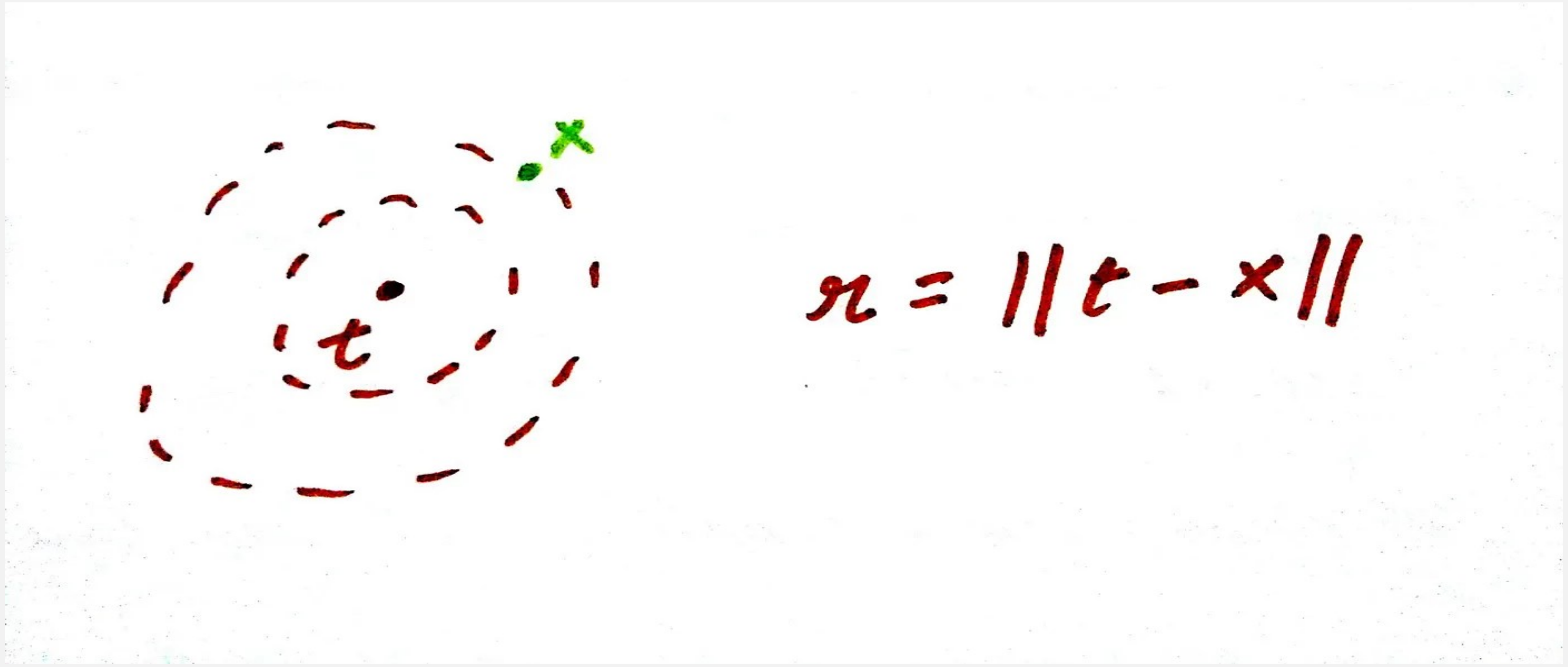
- A Radial Basis Function (RBF) is a function that is only defined by distances from a center.
- Exact position does not matter; only relative position matters.
- RBFN place a radial basis function centered at each point, then perform nonlinear **mapping the data points to higher-dimensional spaces** that are easier to separate.
- Radial Basis Networks are simple two-layer architectures with one layer of **RBF neurons** and one layer of **output neurons**.
- RBF neurons are each assigned a ‘central vector’, from which input vectors are compared.
- These networks fundamentally utilize density and hence are able to model complex nonlinearities with very small structures.

## Radial Basis Function

- A radial basis function (RBF) is a mathematical function that assigns an output value based on the distance between the input data point and a reference point called the center.
- RBFs are commonly used in the hidden layer of RBFN to transform input data from the input space into high-dimensional feature space.
- Every radial basis function has a receptor (t) and input (I) moves radially away from this receptor, the value of the function goes on increasing or goes on decreasing.
- Function value at a point depends upon the radial distance of the point from the receptor t.

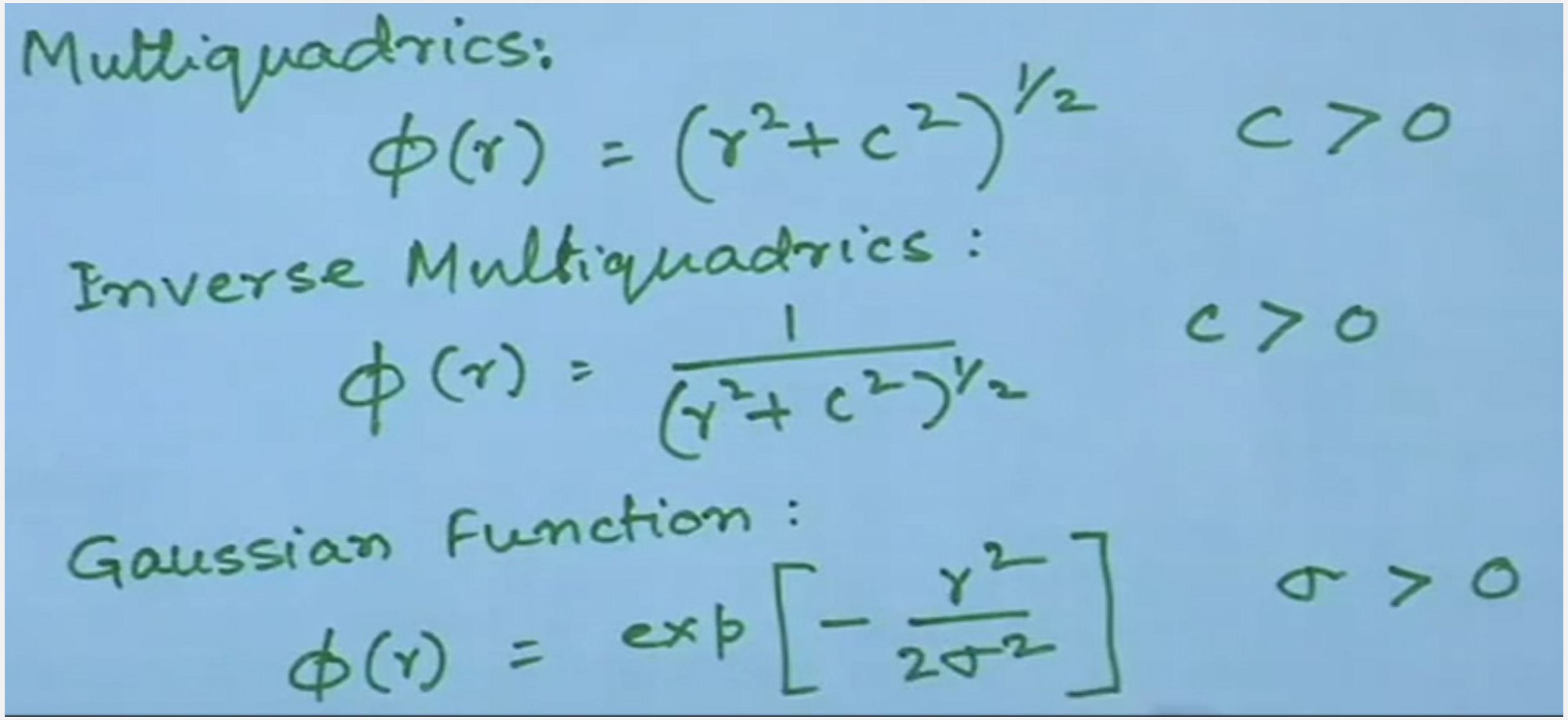
## Radial Basis Function

- If a concentric circles draw around the receptor, the value of the function ($\varphi(x)$) on the concentric circles are constant.



## Different types of RBFs

- There are different choices of radial basis functions (RBFs).



Multiquadrics:

$$\phi(r) = (r^2 + c^2)^{1/2} \qquad c > 0$$

Inverse Multiquadrics:

$$\phi(r) = \frac{1}{(r^2 + c^2)^{1/2}} \qquad c > 0$$

Gaussian Function:

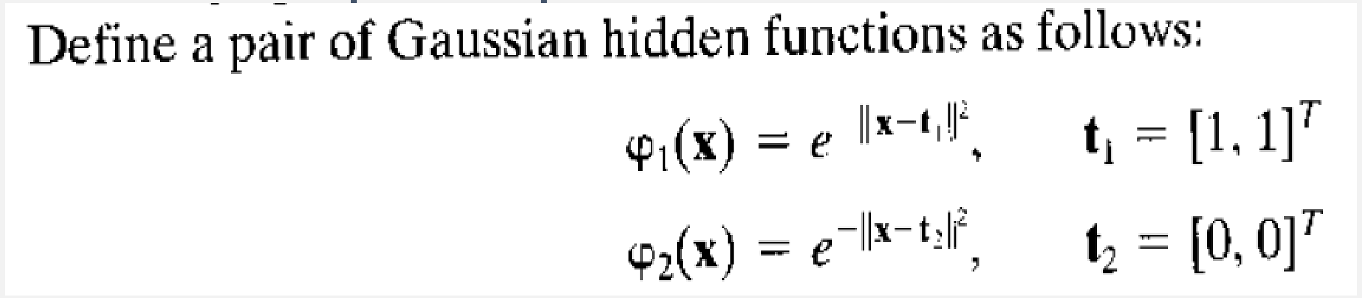
$$\phi(r) = \exp\left[-\frac{r^2}{2\sigma^2}\right] \qquad \sigma > 0$$

## RBFN for Classification

- We apply radial basis function ($\boldsymbol{\varphi}$) to the input feature vector of dimension ($\boldsymbol{n}$) for nonlinear mapping into a higher dimension ($\boldsymbol{m}$) in the hidden layer and being linear mapping in the output layer for classification task.
- When we increase the dimension of the feature vector, the linear separability of feature vector increases.
- The most common radial basis function is Gaussian function.
- **XOR Problem:**
- In XOR problem has four points (patterns or observations):
- (0, 0), (0, 1), (1, 0), and (1, 1) in a 2D space.
- Now, we will design a Classifier using RBFN that produce binary output (y) 0 in response to input patterns (0, 0) and (1, 1) and the binary output 1 in response to input patterns (0, 1) and (1, 0).

## Classification of XOR Problem using RBFN (continued)

- The points that are closest in the input space, in terms of hamming distance, map to regions that are marginally apart in the output space.



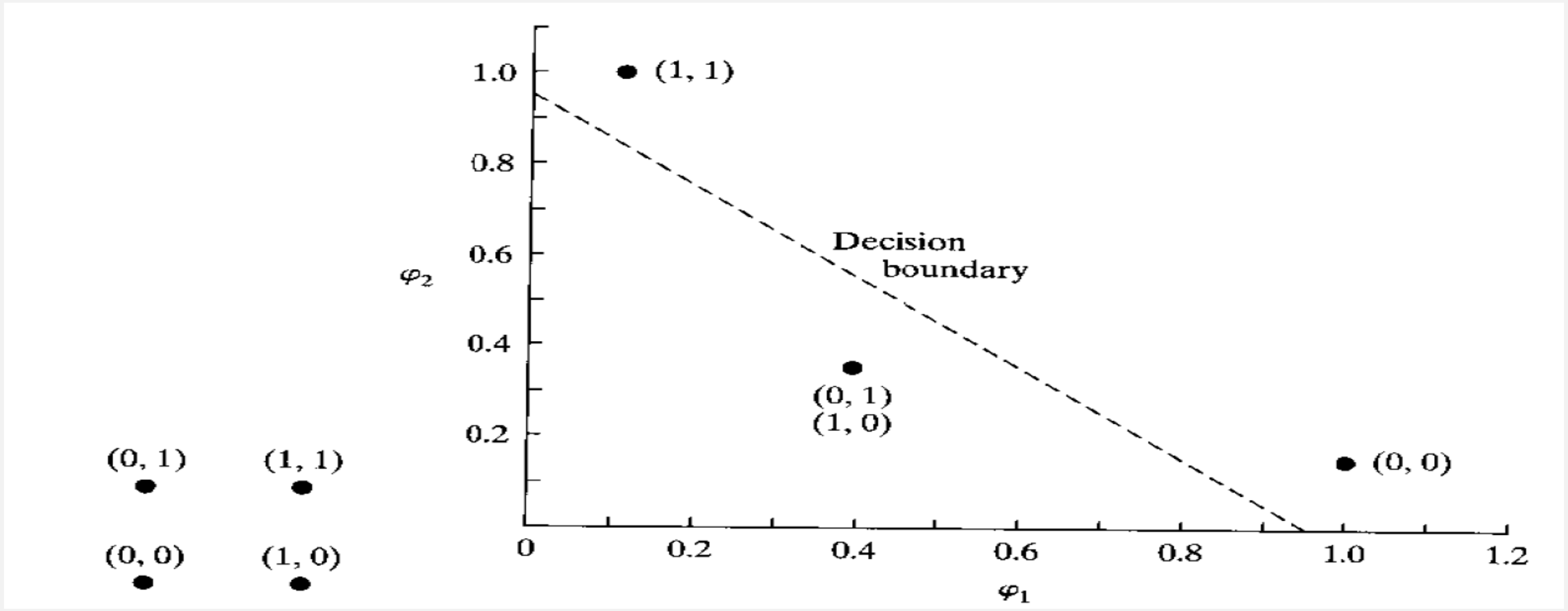
Define a pair of Gaussian hidden functions as follows:

$$\varphi_1(\mathbf{x}) = e^{\|\mathbf{x}-\mathbf{t}_1\|^2}, \qquad \mathbf{t}_1 = [1, 1]^T$$

$$\varphi_2(\mathbf{x}) = e^{-\|\mathbf{x}-\mathbf{t}_2\|^2}, \qquad \mathbf{t}_2 = [0, 0]^T$$

- We will select two receptors $t_1\, and\, t_2$ for non-linear mapping in the hidden layer using Gaussian functions $\varphi_1(x)\, and\, \varphi_1(x)$. Here, the dimension of input and hidden layers are same i.e., 2, still patters are classified linearly in the output layer.
- Output layer of RBFN act as linear classifier such as Perceptron.

- Without increasing the dimensionality of the hidden layer, non-linear Gaussian functions are sufficient to transform the XOR problem into a linearly separable one.



## Input patterns are nonlinearly mapped in hidden layer

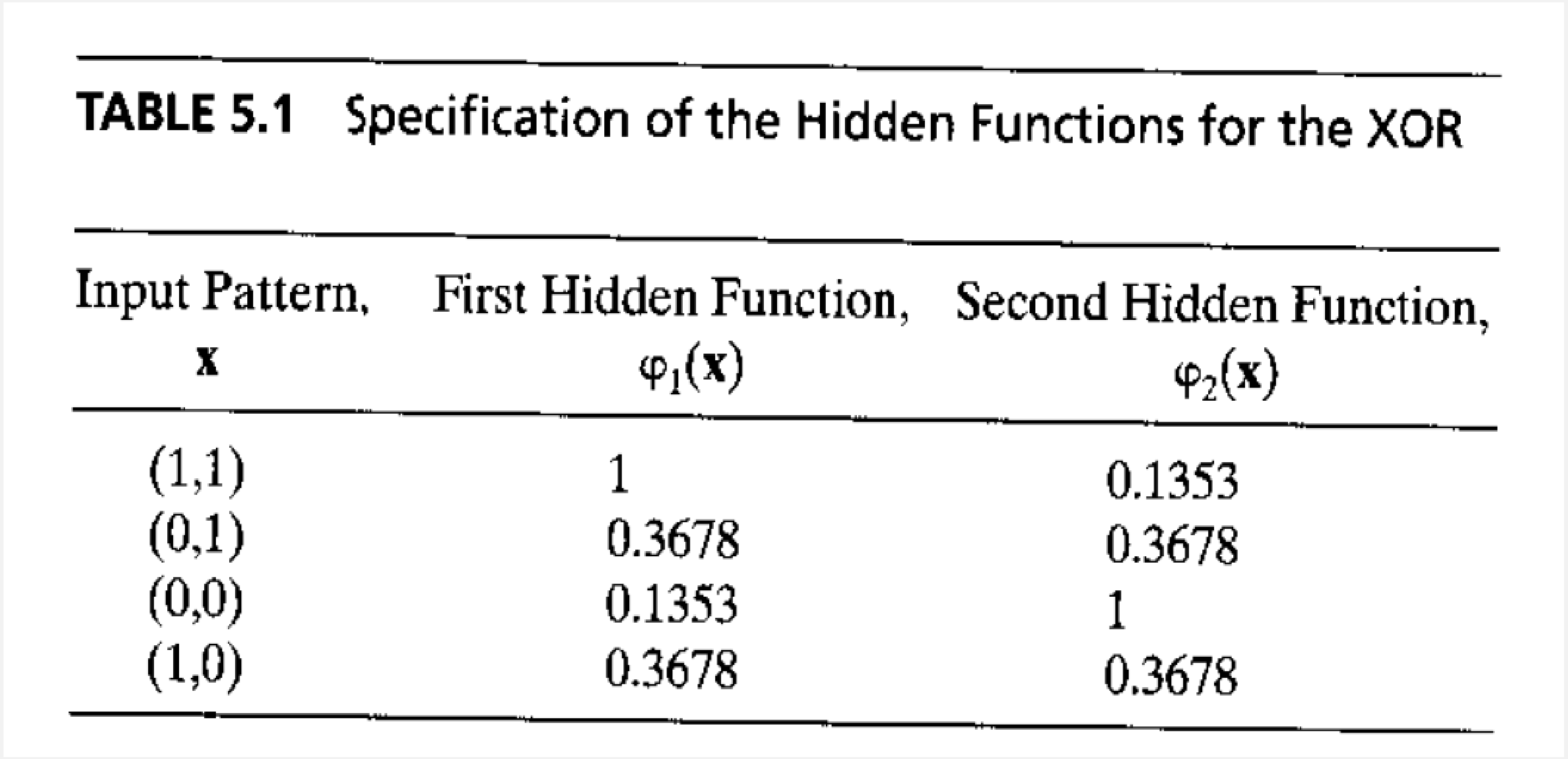
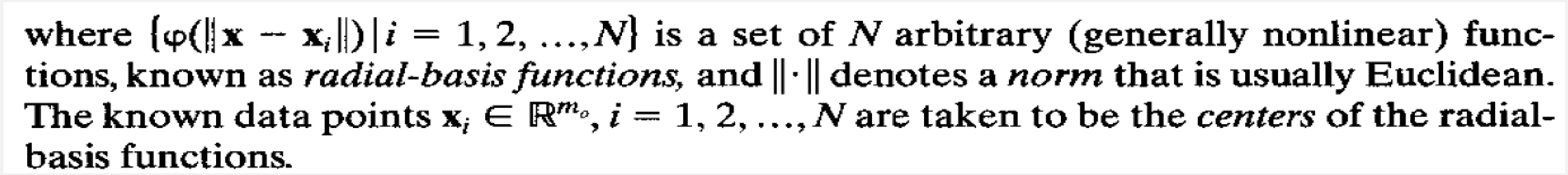


**TABLE 5.1** Specification of the Hidden Functions for the XOR

| Input Pattern, $\mathbf{x}$ | First Hidden Function, $\varphi_1(\mathbf{x})$ | Second Hidden Function, $\varphi_2(\mathbf{x})$ |
|---|---|---|
| (1,1) | 1 | 0.1353 |
| (0,1) | 0.3678 | 0.3678 |
| (0,0) | 0.1353 | 1 |
| (1,0) | 0.3678 | 0.3678 |

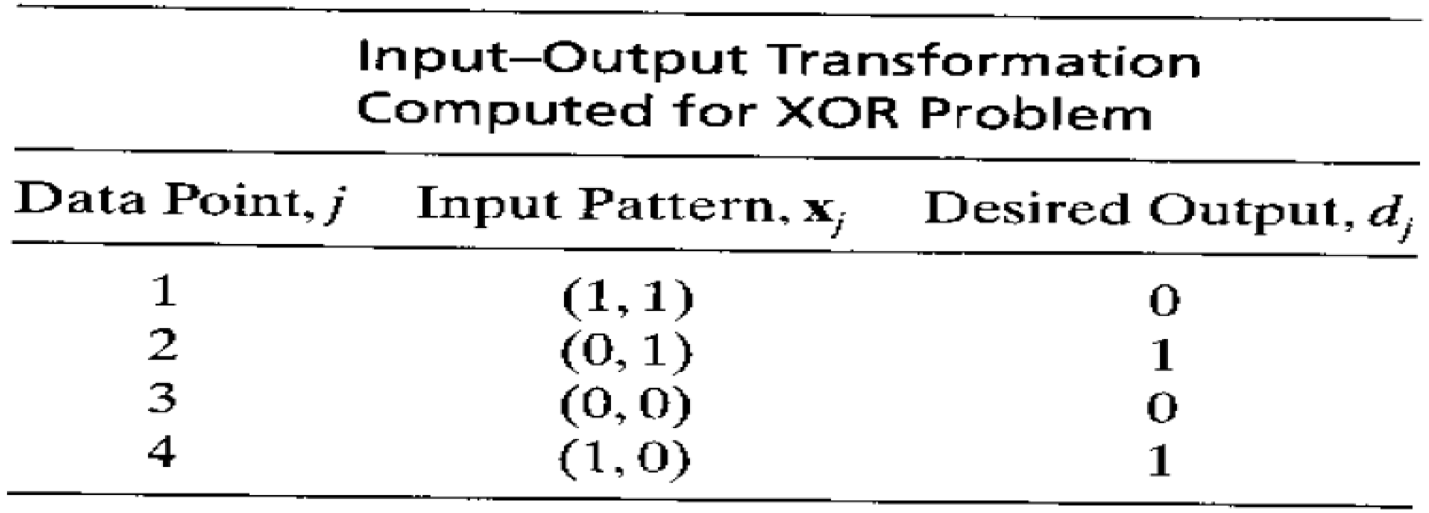
$$y = \sum_{i=1}^{2} w_i \, \varphi(\|x - t_i\|)$$



where $\{\varphi(\|\mathbf{x} - \mathbf{x}_i\|) \mid i = 1, 2, \ldots, N\}$ is a set of $N$ arbitrary (generally nonlinear) functions, known as *radial-basis functions*, and $\|\cdot\|$ denotes a *norm* that is usually Euclidean. The known data points $\mathbf{x}_i \in \mathbb{R}^{m_0}$, $i = 1, 2, \ldots, N$ are taken to be the *centers* of the radial-basis functions.

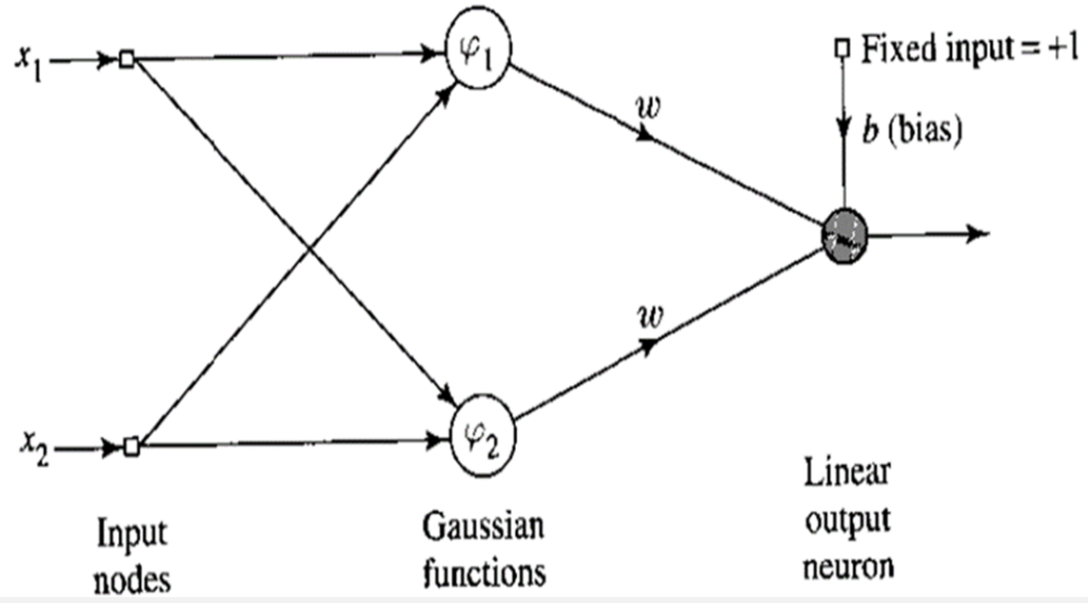
$$\begin{bmatrix} \varphi_{11} & \varphi_{12} \\ \varphi_{21} & \varphi_{22} \\ \varphi_{31} & \varphi_{32} \\ \varphi_{41} & \varphi_{42} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{bmatrix}$$

$$\begin{aligned} w &= \varphi^{+} d \\ &= (\varphi^{T} \varphi)^{-1} \varphi^{T} d \end{aligned}$$



Input–Output Transformation Computed for XOR Problem

| Data Point, $j$ | Input Pattern, $\mathbf{x}_j$ | Desired Output, $d_j$ |
|---|---|---|
| 1 | (1, 1) | 0 |
| 2 | (0, 1) | 1 |
| 3 | (0, 0) | 0 |
| 4 | (1, 0) | 1 |



$$\mathbf{G} = \begin{bmatrix} 1 & 0.1353 & 1 \\ 0.3678 & 0.3678 & 1 \\ 0.1353 & 1 & 1 \\ 0.3678 & 0.3678 & 1 \end{bmatrix}$$

$$\mathbf{d} = [0 \quad 1 \quad 0 \quad 1]^T$$

$$\mathbf{w} = [w \quad w \quad b]^T$$

$$\mathbf{w} = \mathbf{G}^{+}\mathbf{d}$$
$$= (\mathbf{G}^{T}\mathbf{G})^{-1}\mathbf{G}^{T}\mathbf{d}$$

Note that $\mathbf{G}^{T}\mathbf{G}$ is a square matrix with a unique inverse of its own. Substituting Eq. (5.90) in (5.93),we get

$$\mathbf{G}^{+} = \begin{bmatrix} 1.8292 & -1.2509 & 0.6727 & -1.2509 \\ 0.6727 & -1.2509 & 1.8292 & -1.2509 \\ -0.9202 & 1.4202 & -0.9202 & 1.4202 \end{bmatrix}$$

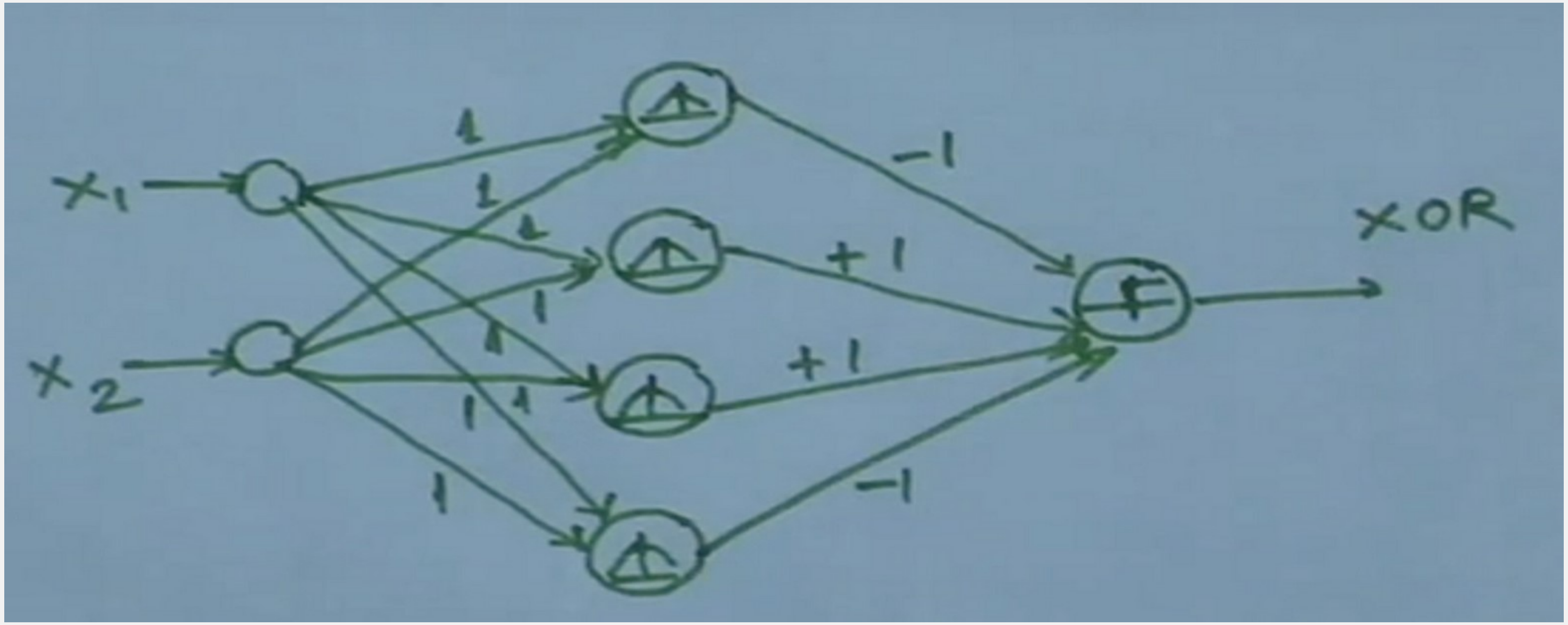
Finally, substituting Eqs. (5.91) and (5.94) in (5.93), we get

$$\mathbf{w} = \begin{bmatrix} -2.5018 \\ -2.5018 \\ +2.8404 \end{bmatrix}$$

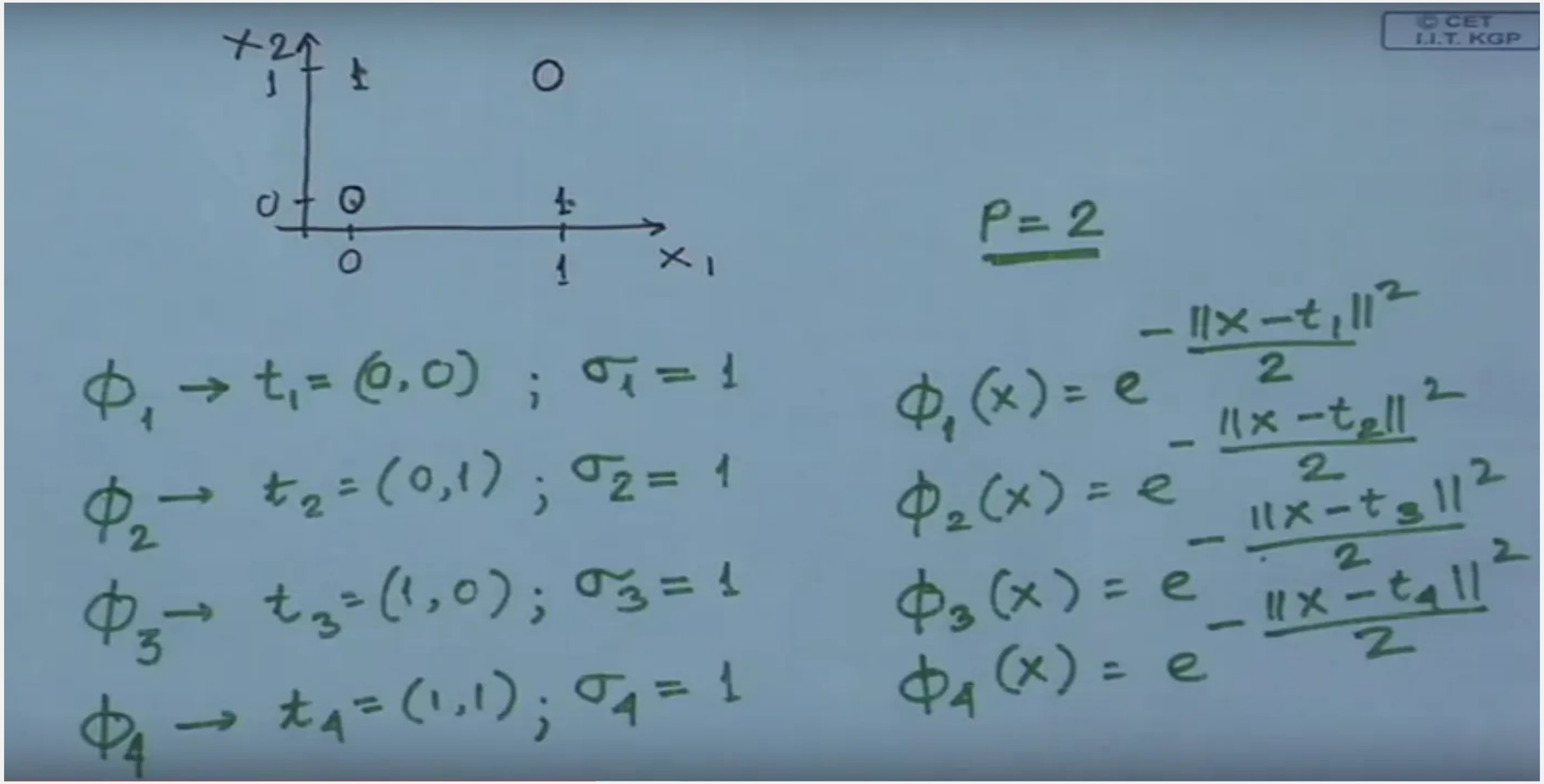
which completes the specification of the RBF network.

# Revisited XOR Problem

- In this case, we will consider four points as receptors. Feature vector has p features. Here, P=2.
- Each node in the hidden layer, performs non-linear transformation using Gaussian radian basis function.



# Transformation Function with receptors



$x_2$
1
0
1
0
0
1
0
1
$x_1$

$P = 2$

$\phi_1 \rightarrow t_1 = (0,0)\ ;\ \sigma_1 = 1$

$\phi_2 \rightarrow t_2 = (0,1)\ ;\ \sigma_2 = 1$

$\phi_3 \rightarrow t_3 = (1,0)\ ;\ \sigma_3 = 1$

$\phi_4 \rightarrow t_4 = (1,1)\ ;\ \sigma_4 = 1$

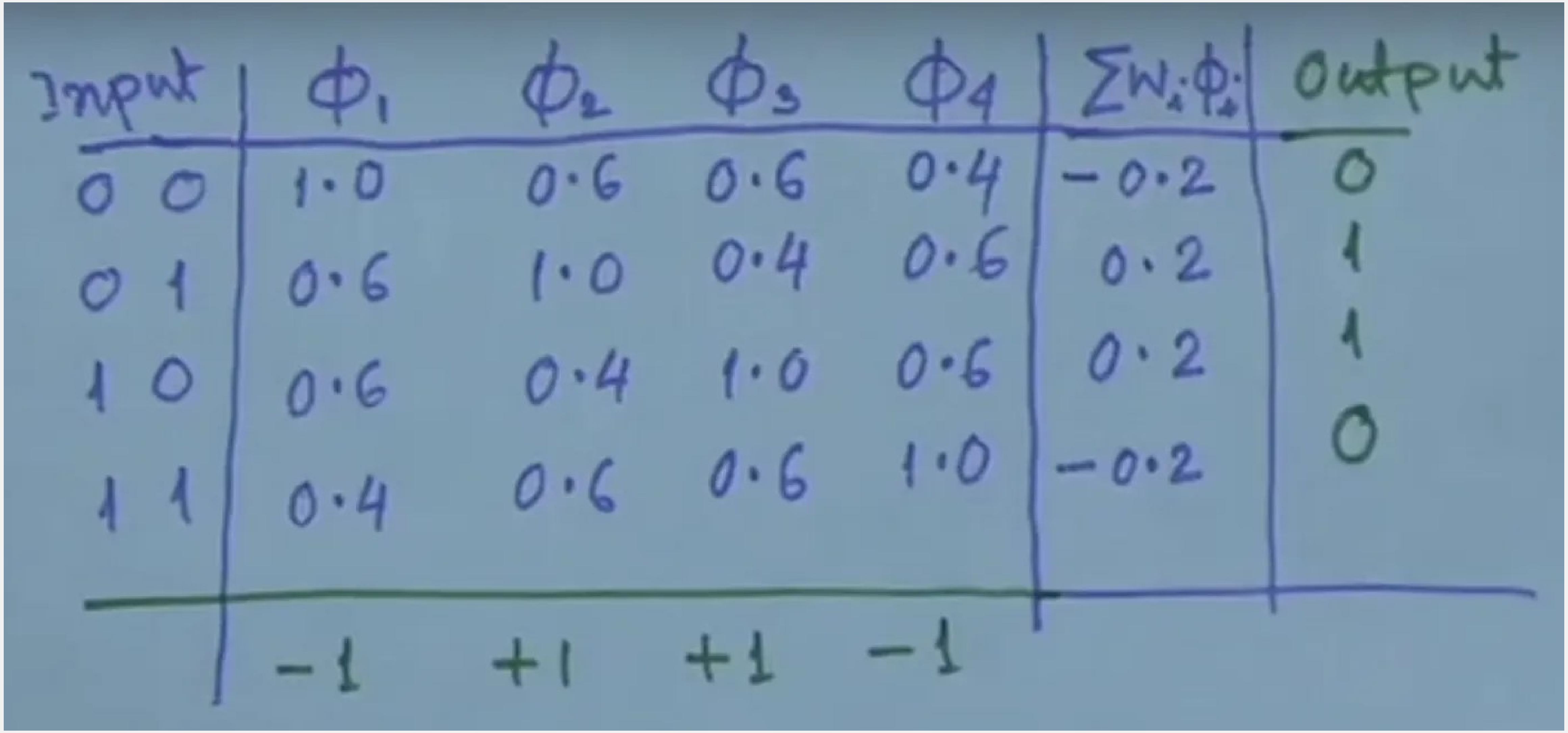
$$\phi_1(x) = e^{-\frac{\|x - t_1\|^2}{2}}$$

$$\phi_2(x) = e^{-\frac{\|x - t_2\|^2}{2}}$$

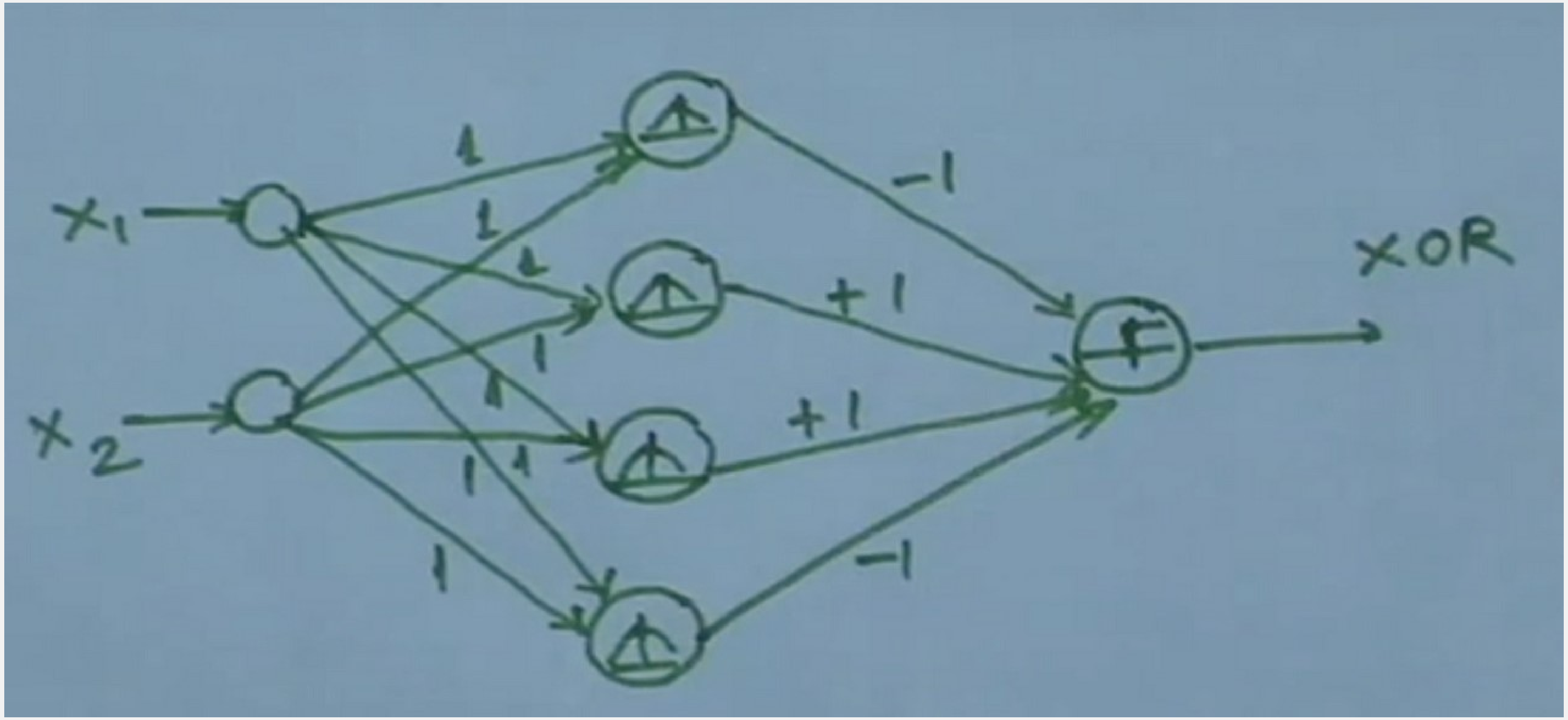
$$\phi_3(x) = e^{-\frac{\|x - t_3\|^2}{2}}$$

$$\phi_4(x) = e^{-\frac{\|x - t_4\|^2}{2}}$$

# Linear Combination of Transformation Function



| Input | $\phi_1$ | $\phi_2$ | $\phi_3$ | $\phi_4$ | $\sum w_i \phi_i$ | Output |
|---|---|---|---|---|---|---|
| 0 0 | 1·0 | 0·6 | 0·6 | 0·4 | −0·2 | 0 |
| 0 1 | 0·6 | 1·0 | 0·4 | 0·6 | 0·2 | 1 |
| 1 0 | 0·6 | 0·4 | 1·0 | 0·6 | 0·2 | 1 |
| 1 1 | 0·4 | 0·6 | 0·6 | 1·0 | −0·2 | 0 |
|  | −1 | +1 | +1 | −1 |  |  |



$x_1$
$x_2$
1
1
1
1
1
1
1
1
-1
+1
+1
-1
XOR

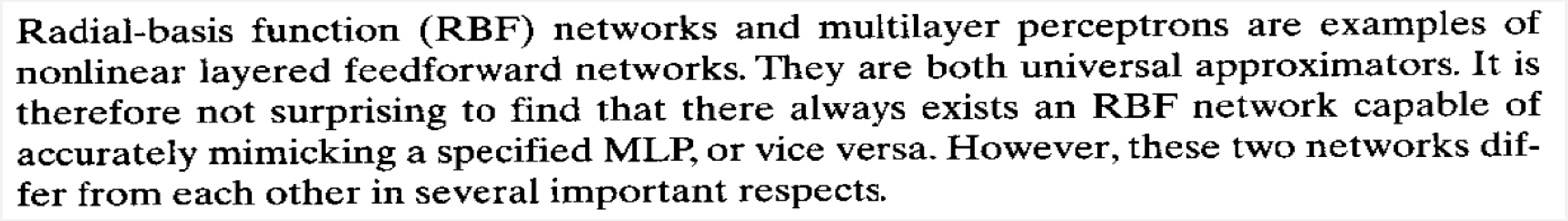
## Advantages

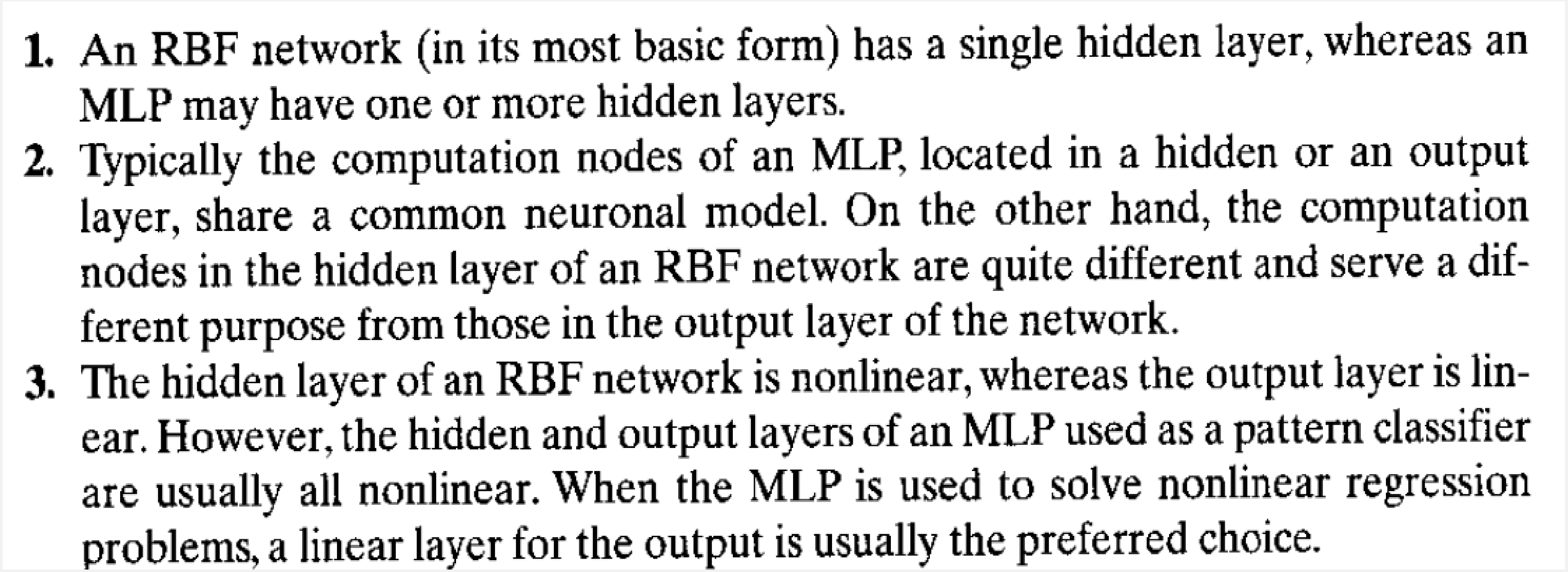
1. In Comparison to MultiLayer Perceptron the training phase is faster as there is no back propagation learning involved.
2. The interpretation roles of Hidden layer nodes are easy as in comparison to multi layer perceptron.
3. Number of hidden layer and number of nodes in hidden layer are decided in case of RBF network but in multi layer perceptron there is no analytical approach to decide the number of nodes in hidden layer or number of hidden layer.
4. Classification task is always performed on linearly separable data only at output layer like Perceptron.

## Disadvantages

- Although the training is faster in RBF network but classification is slow in comparison to Multi layer Perceptron due to fact that every node in hidden layer have to compute the RBF function for the input sample vector during classification.

## Comparison of RBFN and MLP


Radial-basis function (RBF) networks and multilayer perceptrons are examples of nonlinear layered feedforward networks. They are both universal approximators. It is therefore not surprising to find that there always exists an RBF network capable of accurately mimicking a specified MLP, or vice versa. However, these two networks differ from each other in several important respects.


1. An RBF network (in its most basic form) has a single hidden layer, whereas an MLP may have one or more hidden layers.
2. Typically the computation nodes of an MLP, located in a hidden or an output layer, share a common neuronal model. On the other hand, the computation nodes in the hidden layer of an RBF network are quite different and serve a different purpose from those in the output layer of the network.
3. The hidden layer of an RBF network is nonlinear, whereas the output layer is linear. However, the hidden and output layers of an MLP used as a pattern classifier are usually all nonlinear. When the MLP is used to solve nonlinear regression problems, a linear layer for the output is usually the preferred choice.

4. The argument of the activation function of each hidden unit in an RBF network computes the *Euclidean norm* (*distance*) between the input vector and the center of that unit. Meanwhile, the activation function of each hidden unit in an MLP computes the *inner product* of the input vector and the synaptic weight vector of that unit.
5. MLPs construct *global* approximations to nonlinear input–output mapping. On the other hand, RBF networks using exponentially decaying localized nonlinearities (e.g., Gaussian functions) construct *local* approximations to nonlinear input–output mappings.

This in turn means that for the approximation of a nonlinear input–output mapping, the MLP may require a smaller number of parameters than the RBF network for the same degree of accuracy.