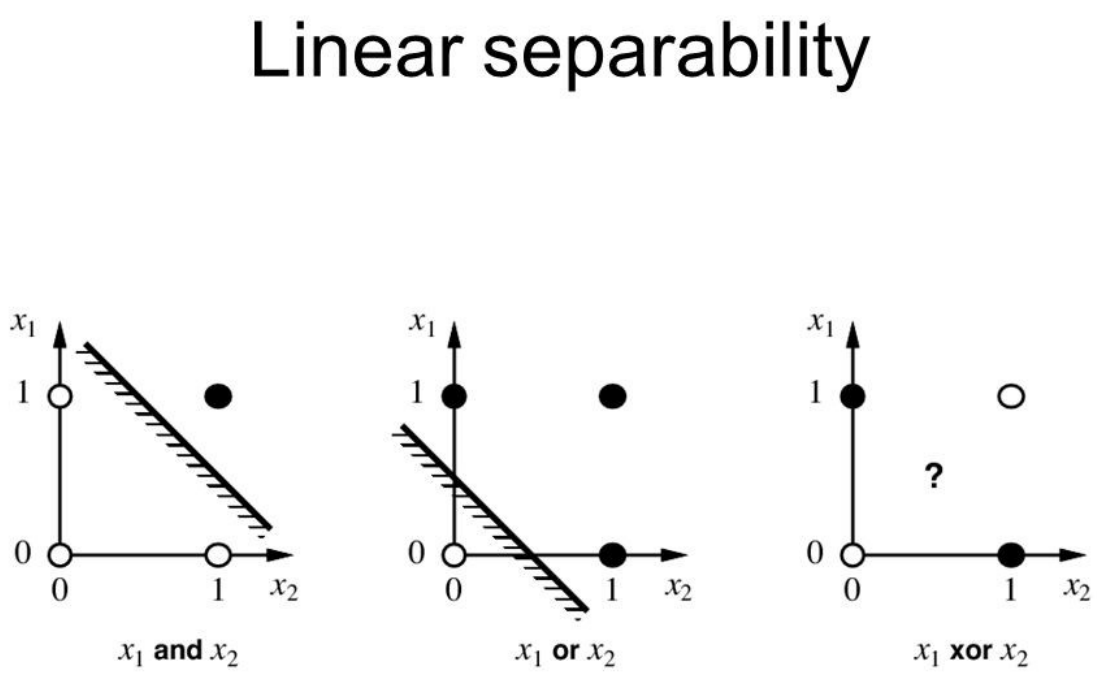
When single layer perceptron is sufficient and when do we need hidden layers?

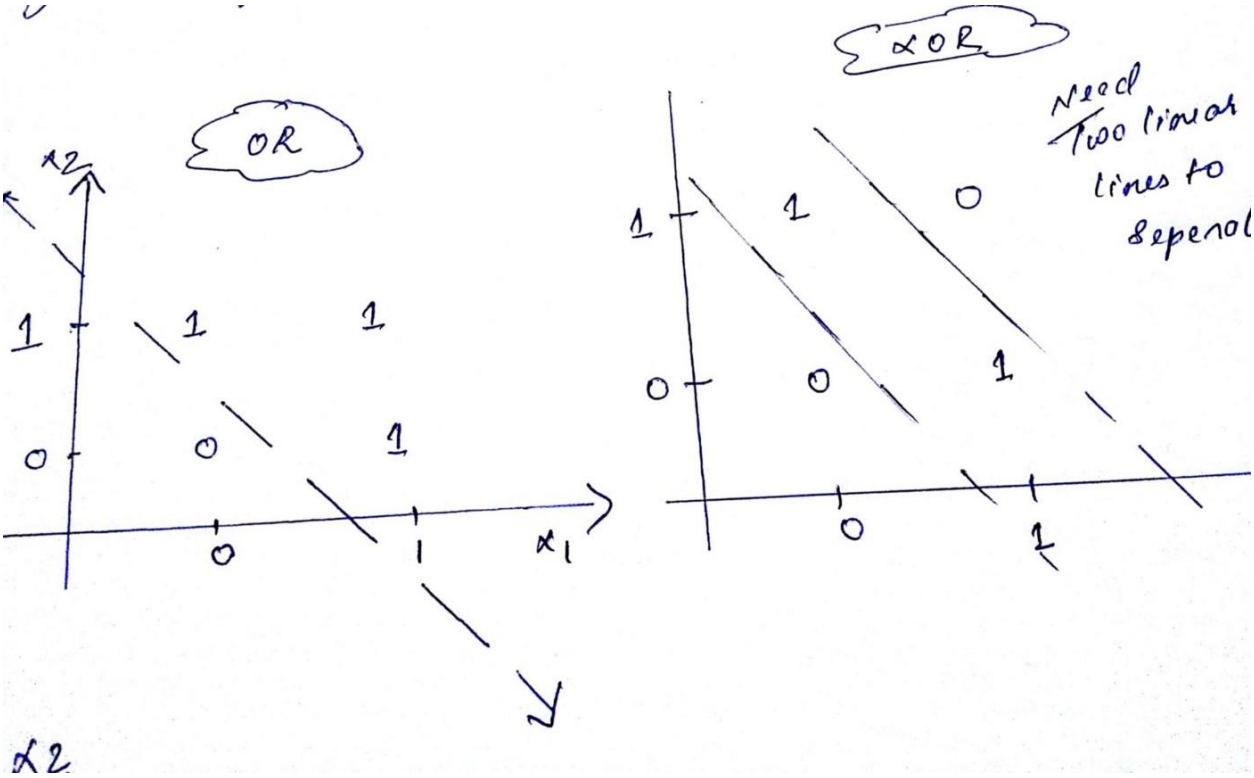
If the classes or patterns are linearly separable then single layer Perceptron is sufficient otherwise we need to incorporate hidden layers in the network to introduce non-linearity in the network.

The hidden layer simply represents non-linear boundaries by a set of pairwise linear boundaries.

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.

Let's look at an example of OR and XOR gates output. We can see OR output can be linearly classified but the same is not true with XOR outputs as we need two linear lines to separate the boundaries of output (0 and 1).

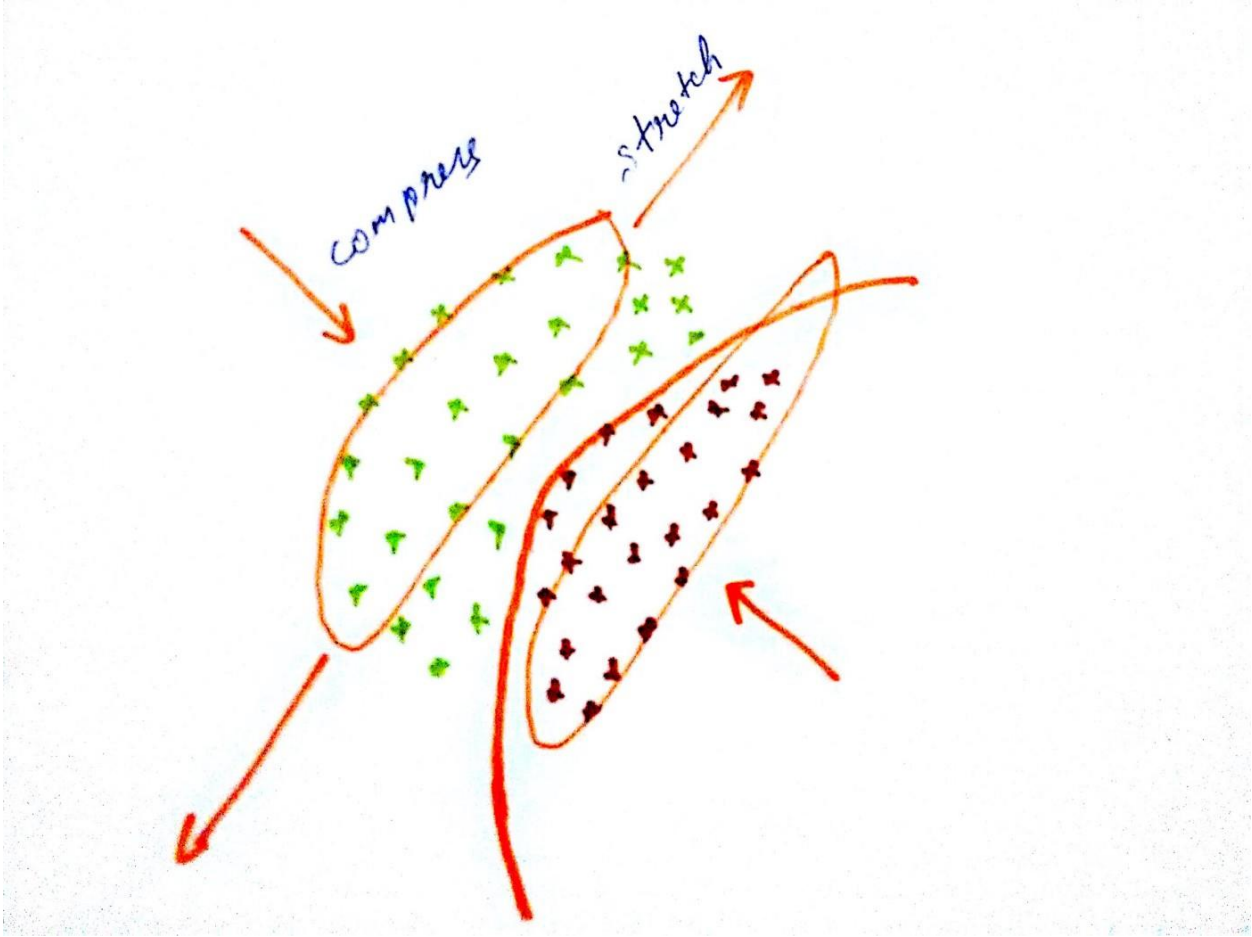




What is Radial Basis Function?

So coming to Radial Basis Function (RBF) what it does for our above problem of non-linear separable patterns. RBF performs nonlinear transformation over input vectors before they are fed for classification with help of the below transformations.

a) Imposes non-linear transformation on input feature vector.
b) Increases Dimensionality of the feature vector.



In above image green and red is the feature vector of two different classes which are quite evident that they are non linearly separable so once we impose RBF what it does that it performs a non-linear transformation on feature vectors which in effect compress them along the width and stretches them along length due to which it becomes linearly separable.

Along with the non-linear transformation, the dimensionality is increased up to the M dimension from the P original dimension with the below equations.

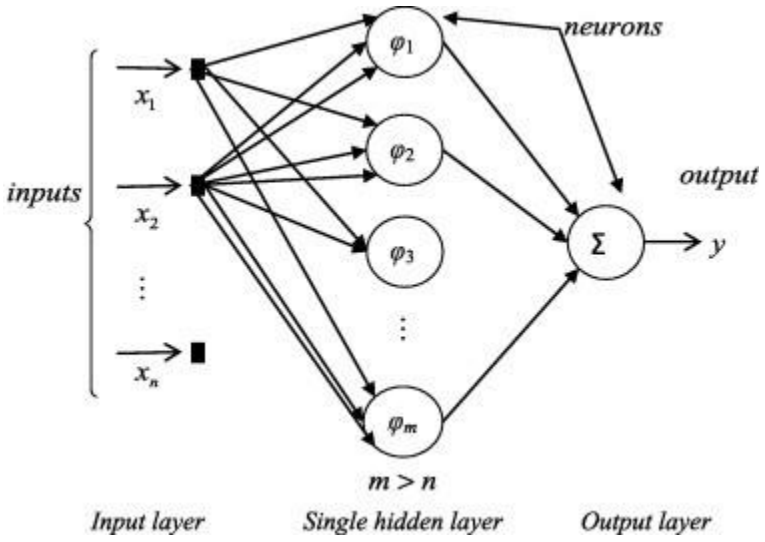
$p \rightarrow M$

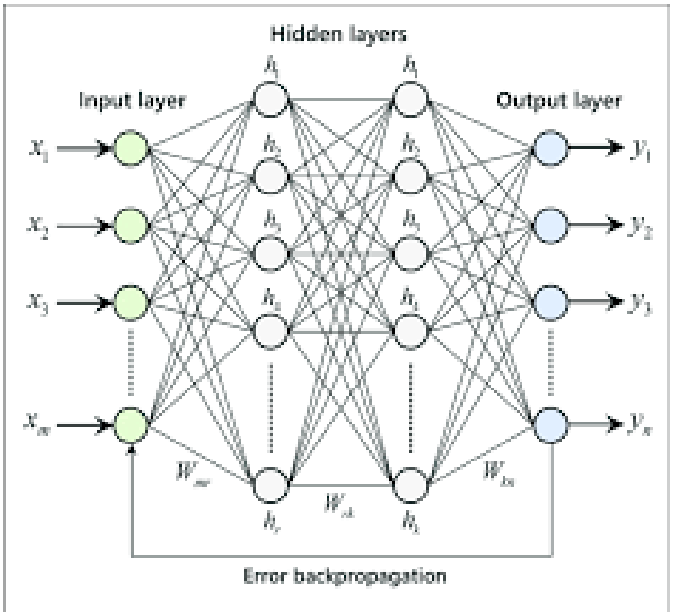
$M > p$

RBF transforms p dim to M dim

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

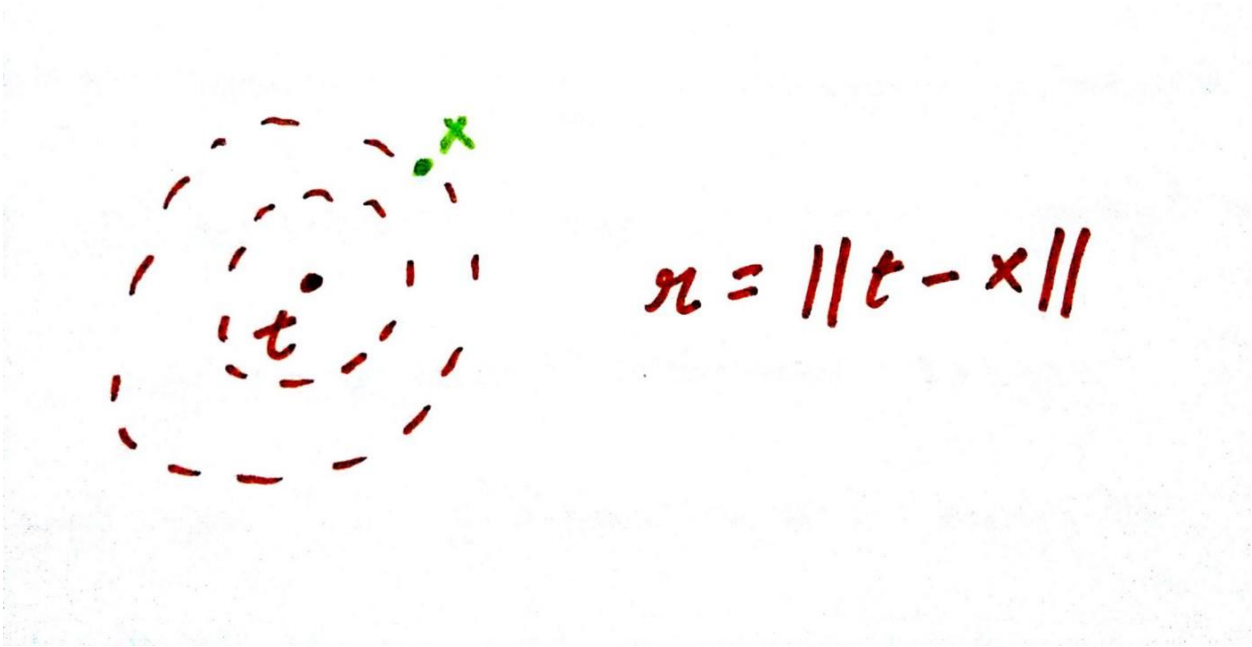
all values of ϕ are real





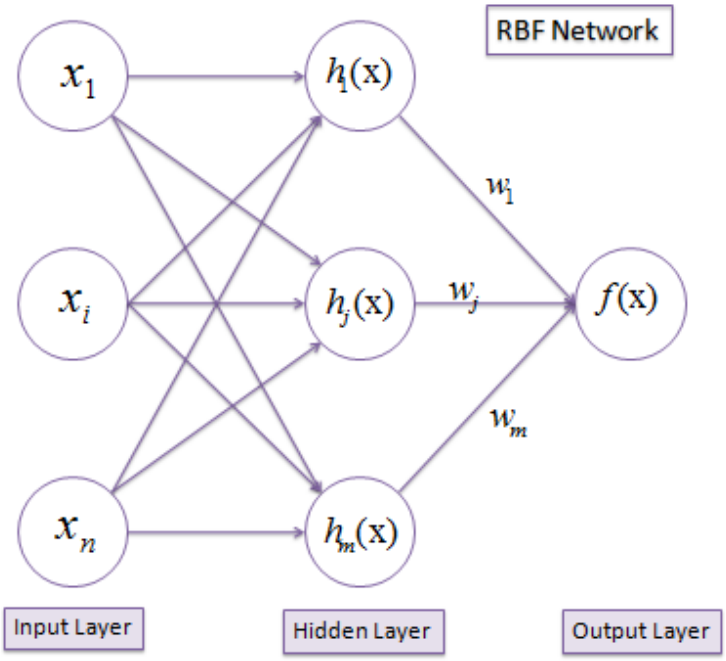
Explain the RBF behaviour.

Each RBF function has a receptor and values of the function either increases or decreases while moving away from receptor t.



r is the distance between receptor t and any input feature vector X

Compiled By: Dr Md Ashraf Uddin, PhD(Federation University Australia)



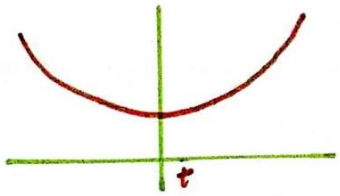
$$f(\mathrm{x}) = \sum_{j=1}^{m} w_j h_j(\mathrm{x})$$

$$h(x) = \exp\left(-\frac{(x-c)^2}{r^2}\right)$$

Choices of RBF :-

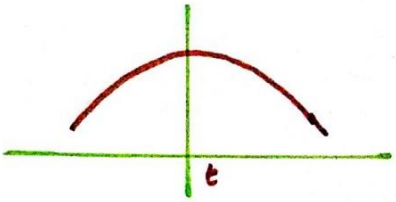
(1) MULTIQUADRICS :

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



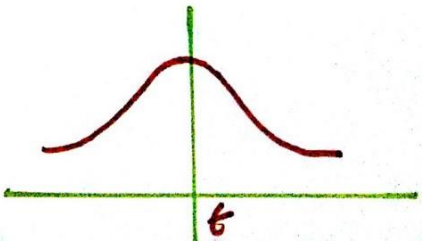
(2) INVERSE MULTIQUADRICS :

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



(3) GAUSSIAN FUNCTION :

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



Commonly used types of radial basis functions include (writing $r = \|\mathbf{x} - \mathbf{x}_i\|$ and using ε to indicate a *shape parameter* that can be used to scale the input of the radial kernel[11]):

- Infinitely Smooth RBFs

These radial basis functions are from $C^\infty(\mathbb{R})$ and are strictly positive definite functions[12] that require tuning a shape parameter ε

- Gaussian:

$$\varphi(r) = e^{-(\varepsilon r)^2}, \qquad (2)$$

- Multiquadric:

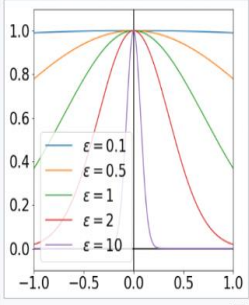
$$\varphi(r) = \sqrt{1 + (\varepsilon r)^2}, \qquad (3)$$

- Inverse quadratic:

$$\varphi(r) = \frac{1}{1 + (\varepsilon r)^2}, \qquad (4)$$

- Inverse multiquadric:

$$\varphi(r) = \frac{1}{\sqrt{1 + (\varepsilon r)^2}}, \qquad (5)$$

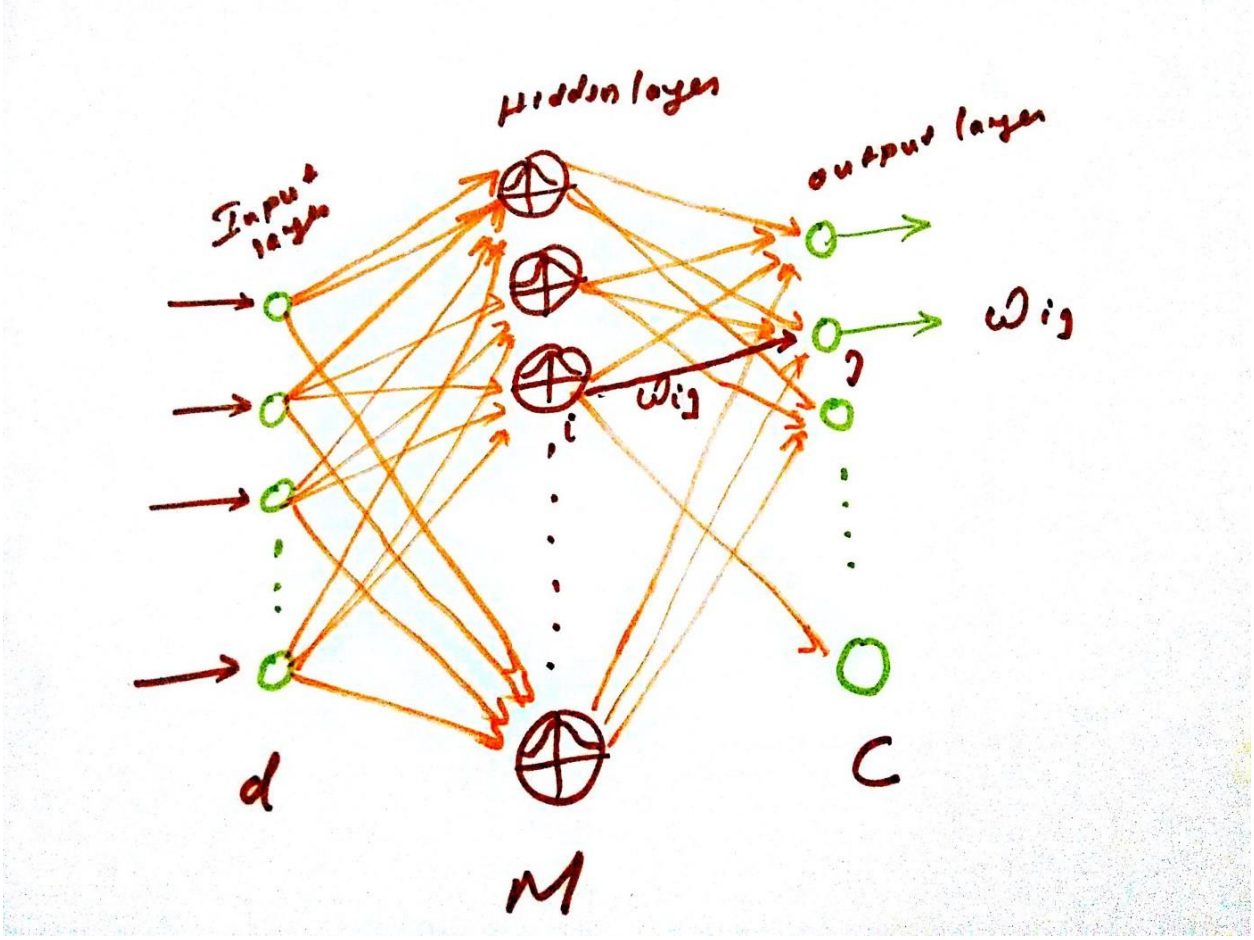


Gaussian function for several choices of ε

Above are the general choice for RBF but gaussian is the most popular one, we can see the function decays while going away from the receptor.

The network architecture of RBF

Radial basis function (RBF) networks typically have three layers: an input layer, a hidden layer with a non-linear RBF activation function and a linear output layer. The input can be modelled as a vector of real numbers. The output of the network is then a scalar function of the input vector.



d is the dimensionality of input feature space, M is the dimensionality of transformed feature space where we have imposed our RBF, C is the number of classes to be identified.

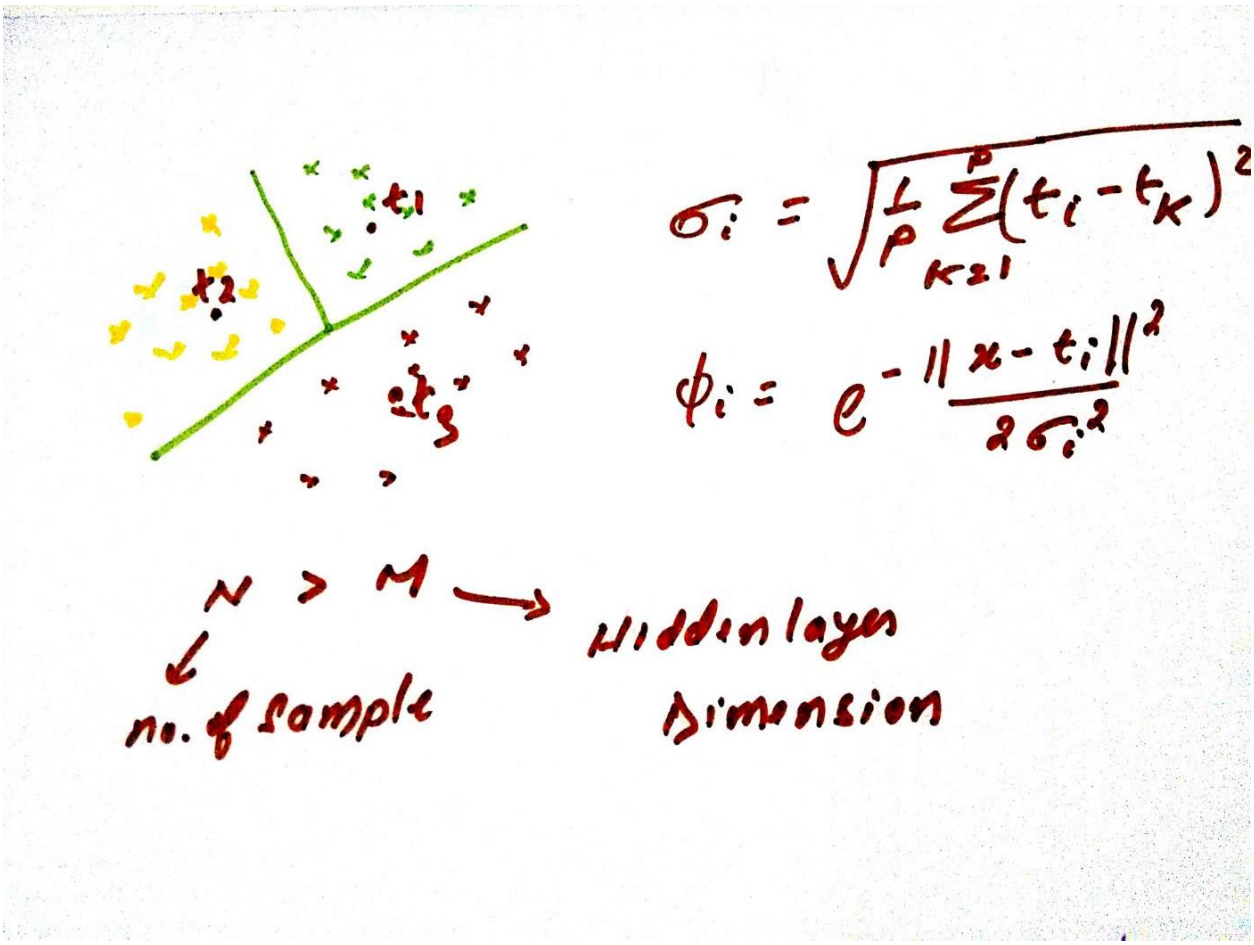
Training Comprises for this kind of network in two phases:

a) Training Hidden layer which comprises of M RBF functions, the parameters to be determined for RBF function are receptor position t and the Sigma in case of Gaussian RBF.
b) Training weight vectors Wij for the output layer.

Training Hidden layer:

So for training hidden layers, there are different approaches, let us assume for now we are dealing with Gaussian RBF so we need to determine receptor t and Spread ie Sigma. One of the approaches is to randomly select the M number of the receptor from the N number of sample feature vectors but this does not seem logical so we can go ahead with a clustering mechanism to determine receptors ti.

As we have M nodes in the hidden layer and N samples so for clustering to work here N>M.



Calculation of receptors:

Let's look at the above example where we have M=3 so we need to determine three t's. so initially we divide out feature vector space into three arbitrary clusters and took their means as the initial receptors, then we need to iterate for every sample feature vector and perform the below steps:

a) From the selected input feature vector x determine distances of means(t1,t2,t3) of three different clusters whichever distance mean is minimum sample x will get assigned to that cluster.
b) After x got assigned to different cluster all the means(t1,t2,t3) gets recomputed
c) Perform step 1 and step 2 for all sample points

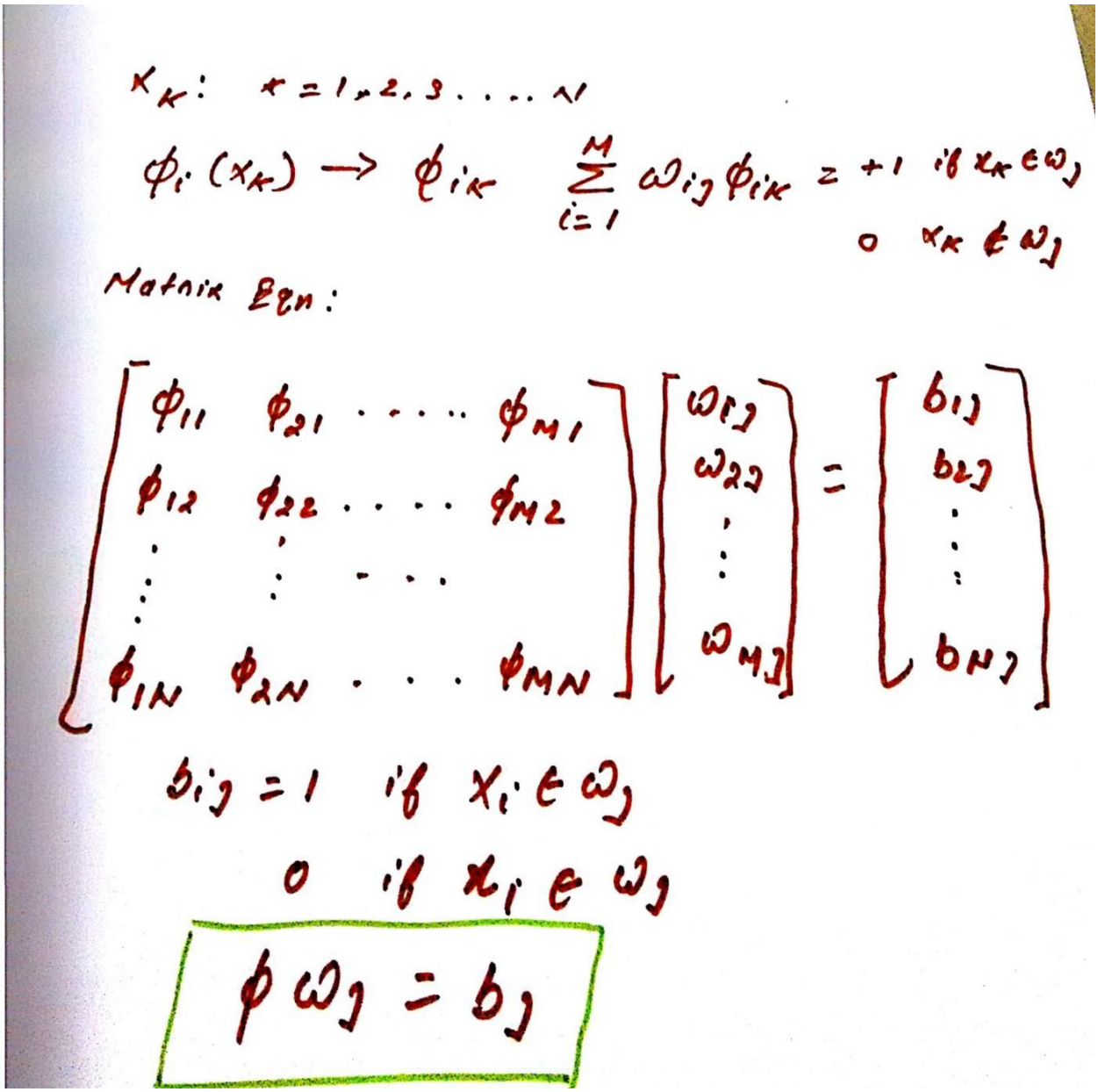
Once the iteration finishes we will get the optimal t1,t2 and t3.

Calculation of Sigma:

once receptors are calculated we can use K nearest neighbour algorithm to calculate sigma the formula is there in the above image. we need to select the values of P.

Training Weight Vectors

Let us assume the dimensionality of the hidden layer as M and sample size as N we can calculate the optimal weight vector for the network using the pseudo-inverse matrix solution.



Let us assume we don't have a perfect solution so we will take the error e and try to optimize our criteria function J(wj) using a closed-form solution to get our optimum Wj.

$$E = \frac{1}{2} * \sum (t - o)^2 \quad \text{(error)}$$

$$\frac{\partial E}{\partial w_{jk}} \text{ (gradient)}$$

$$\Delta w_{jk} = -1 * \eta * [\, h_j * (o_k - t_k) * o_k * (1 - o_k) \,]$$

learn rate

derivative of error

derivative of softmax

The above Wj calculation can be done for every output node j =1,2….C to get the respective weight vectors for them.

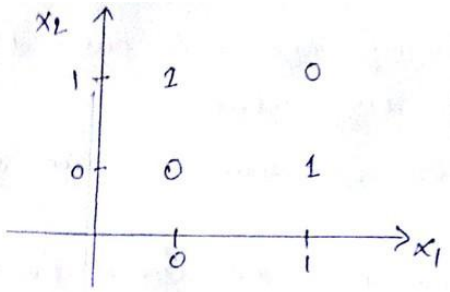
Advantages of RBF

1) In Comparison, to MultiLayer Perceptron the training phase is faster as there is no backpropagation learning involved.
2) The interpretation roles of Hidden layer nodes are easy in comparison to multi-layer perceptron.
3) The number of hidden layers and the number of nodes in the hidden layer are decided in the case of the RBF network but in multi-layer perceptron, there is no analytical approach to decide the number of nodes in the hidden layer or number of the hidden layers.

Disadvantages of RBF

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

Example: Solution for XOR gate using RBF Network



$P = 2$

Radial basis function

$\phi_1 \rightarrow t_1 = (0,0)$; $\sigma_1 = 1$ $\quad \phi_1(x) = e^{-\frac{\|x - t_1\|^2}{2}}$

$\phi_2 \rightarrow t_2 = (0,1)$; $\sigma_2 = 1$ $\quad \phi_2(x) = e^{-\frac{\|x - t_2\|^2}{2}}$

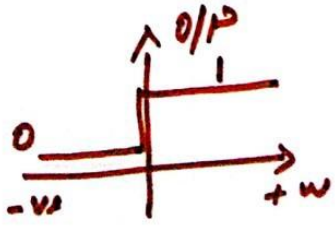
$\phi_3 \rightarrow t_3 = (1,0)$; $\sigma_3 = 1$ $\quad \phi_3(x) = e^{-\frac{\|x - t_3\|^2}{2}}$

$\phi_4 \rightarrow t_4 = (1,1)$; $\sigma_4 = 1$ $\quad \phi_4(x) = e^{-\frac{\|x - t_4\|^2}{2}}$

\# converting 2 dim to 4 dim

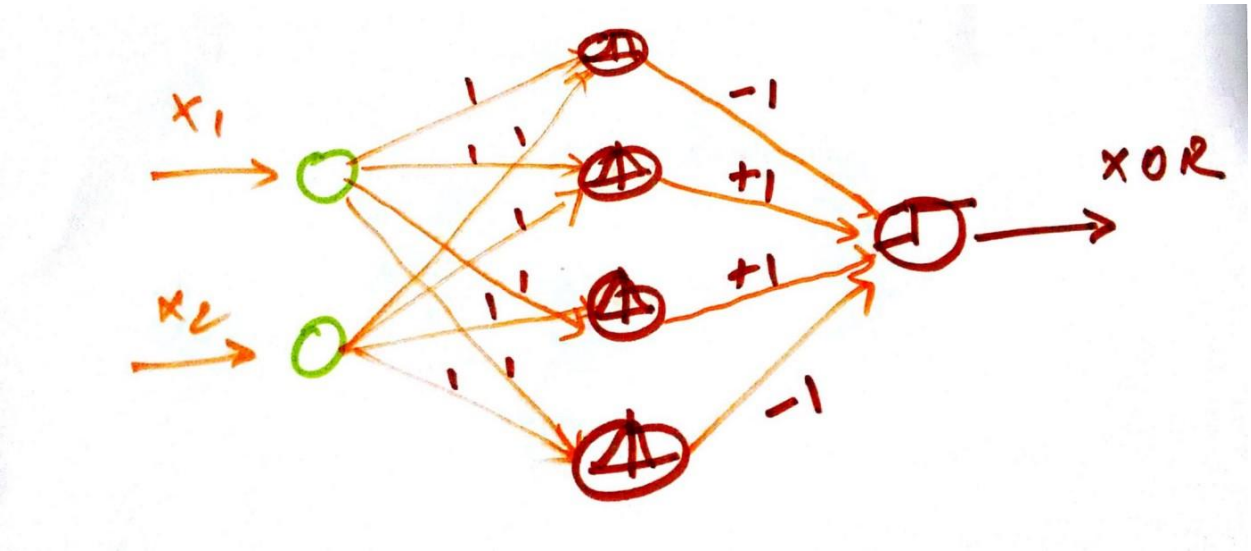
Input	ϕ_1	ϕ_2	ϕ_3	ϕ_4	$\Sigma w_i \phi_i$	output
0 0	1.0	0.6	0.6	0.4	-0.2	0
0 1	0.6	1.0	0.4	0.6	0.2	1
1 0	0.6	0.4	1.0	0.6	0.2	1
1 1	0.4	0.6	0.6	1.0	-0.2	0
weights →	-1	+1	+1	-1		

Hard threshold



Here optimal weights are -1,+1,+1 and -1 and at the output, we have enforced hard thresholding function as if the value is -ve o/p is 0 if it's +ve o/p is 1

Architecture diagram for XOR



References

https://www.youtube.com/watch?v=KiVJkqac82Q&t=1891s&ab_channel=nptelhrd
https://www.youtube.com/watch?v=nOt_V7ndmLE&t=1669s&ab_channel=nptelhrd
https://medium.com/geekculture/radial-basis-function-network-952785127d61

Implementation

https://www.kaggle.com/residentmario/radial-basis-networks-and-custom-keras-layers/notebook

https://towardsdatascience.com/most-effective-way-to-implement-radial-basis-function-neural-network-for-classification-problem-33c467803319

Compiled By: Dr Md Ashraf Uddin, PhD(Federation University Australia)