Neural Networks

Radial Basis Function Networks

Laxmidhar Behera

Department of Electrical Engineering
Indian Institute of Technology, Kanpur

Radial Basis Function Networks

Radial Basis Function Networks (RBFN) consists of 3 layers

- an input layer
- a hidden layer
- an output layer

The hidden units provide a set of functions that constitute an arbitrary basis for the input patterns.

- hidden units are known as radial centers and represented by the vectors $c_1, c_2, \cdots, c_h$
- transformation from input space to hidden unit space is nonlinear whereas transformation from hidden unit space to output space is linear
- dimension of each center for a p input network is $p \times 1$

Network Architecture

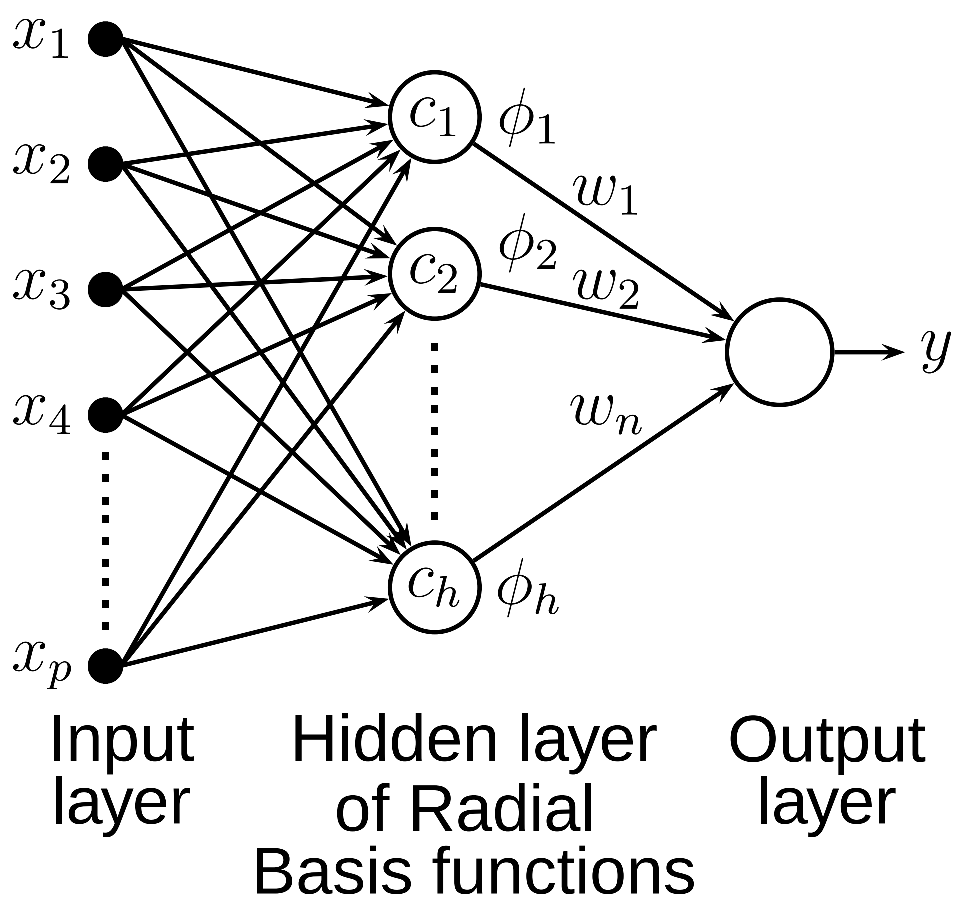


Figure 1: Radial Basis Function Network

Radial Basis functions

The radial basis functions in the hidden layer produces a significant non-zero response only when the input falls within a small localized region of the input space.

Each hidden unit has its own receptive field in input space.

An input vector x_i which lies in the receptive field for center c_j, would activate c_j and by proper choice of weights the target output is obtained. The output is given as

$$y = \sum_{j=1}^{h} \phi_j w_j, \qquad \phi_j = \phi(\|x - c_j\|)$$

w_j: weight of j^{th} center, ϕ: some radial function.

Contd...

Different radial functions are given as follows.

Gaussian radial function	$\phi(z) = e^{-z^2/2\sigma^2}$
Thin plate spline	$\phi(z) = z^2 log z$
Quadratic	$\phi(z) = (z^2 + r^2)^{1/2}$
Inverse quadratic	$\phi(z) = \frac{1}{(z^2+r^2)^{1/2}}$

Here, $z = \|x - c_j\|$

The most popular radial function is Gaussian activation function.

RBFN vs. Multilayer Network

RBF NET	MULTILAYER NET
It has a single hidden layer	It has multiple hidden layers
The basic neuron model as well as the function of the hidden layer is different from that of the output layer	The computational nodes of all the layers are similar
The hidden layer is nonlinear but the output layer is linear	All the layers are nonlinear

Contd...

RBF NET	MULTILAYER NET
Activation function of the hidden unit computes the Euclidean distance between the input vector and the center of that unit	Activation function computes the inner product of the input vector and the weight of that unit
Establishes local mapping, hence capable of fast learning	Constructs global approximations to I/O mapping
Two-fold learning. Both the centers (position and spread) and the weights have to be learned	Only the synaptic weights have to be learned

Learning in RBFN

Training of RBFN requires optimal selection of the parameters vectors c_i and w_i, $i = 1, \cdots h$.

Both layers are optimized using different techniques and in different time scales.

Following techniques are used to update the weights and centers of a RBFN.

- Pseudo-Inverse Technique (Off line)
- Gradient Descent Learning (On line)
- Hybrid Learning (On line)

Pseudo-Inverse Technique

- This is a least square problem. Assume a fixed radial basis functions e.g. Gaussian functions.
- The centers are chosen randomly. The function is normalized i.e. for any x, $\sum \phi_i = 1$.
- The standard deviation (width) of the radial function is determined by an adhoc choice.

The learning steps are as follows:

1. The width is fixed according to the spread of centers

$$\phi_i = e^{(-\frac{h}{d^2}\|x-c_i\|^2)}, \quad i = 1, 2, \cdots h$$

where h: number of centers, d: maximum distance between the chosen centers. Thus $\sigma = \frac{d}{\sqrt{2h}}$.

Contd...

2. From figure 1, $\mathbf{\Phi} = [\phi_1, \phi_2, \cdots, \phi_h]$

$$\begin{aligned} \mathbf{w} &= [w_1, w_2, \cdots, w_h]^T \\ \mathbf{\Phi w} &= y^d, \quad y^d \text{ is the desired output} \end{aligned}$$

3. Required weight vector is computed as

$$\mathbf{w} = (\mathbf{\Phi}^T \mathbf{\Phi})^{-1} \mathbf{\Phi}^T y^d = \mathbf{\Phi}' y^d$$

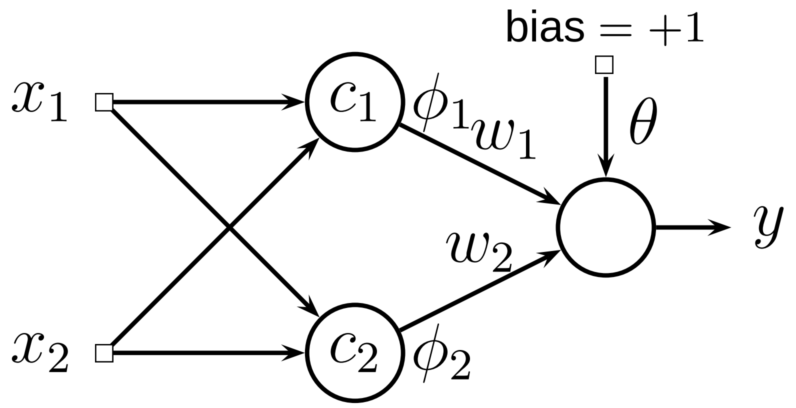
$\mathbf{\Phi}' = (\mathbf{\Phi}^T \mathbf{\Phi})^{-1} \mathbf{\Phi}^T$ is the pseudo-inverse of $\mathbf{\Phi}$.

This is possible only when $\mathbf{\Phi}^T \mathbf{\Phi}$ is non-singular. If this is singular, singular value decomposition is used to solve for $\mathbf{w}$.

Illustration: EX-NOR problem

The truth table and the RBFN architecture are given below:

x_1	x_2	y^d
0	0	1
0	1	0
1	0	0
1	1	1



Choice of centers is made randomly from 4 input patterns.

$$c_1 = [0\ 0]^T \text{ and } c_2 = [1\ 1]^T$$

$$\phi_1 = \phi(\|x - c_1\|) = e^{-\|x-c_1\|^2}$$

Similarly, $\phi_2 = e^{-\|x-c_2\|^2}, \quad x = [x_1\ x_2]^T$

Contd...

Output $y = w_1\phi_1 + w_2\phi_2 + \theta$

Applying 4 training patterns one after another

$$w_1 + w_2 e^{-2} + \theta = 1 \qquad w_1 e^{-1} + w_2 e^{-1} + \theta = 1$$
$$w_1 e^{-1} + w_2 e^{-1} + \theta = 1 \qquad w_1 e^{-2} + w_2 + \theta = 1$$

$$\mathbf{\Phi} = \begin{bmatrix} 1 & 0.1353 & 1 \\ 0.3679 & 0.3679 & 1 \\ 0.3679 & 0.3679 & 1 \\ 0.1353 & 1 & 1 \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \theta \end{bmatrix}, \quad y^d = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Using $\mathbf{w} = \mathbf{\Phi}' y^d$, we get $\mathbf{w} = \begin{bmatrix} 2.5031 & 2.5031 & -1.848 \end{bmatrix}^T$

Gradient Descent Learning

One of the most popular approaches to update c and w, is supervised training by error correcting term which is achieved by a gradient descent technique. The update rule for center learning is

$$c_{ij}(t+1) = c_{ij}(t) - \eta_1 \frac{\partial E}{\partial c_{ij}}, \quad \text{for } i = 1 \text{ to } p, \quad j = 1 \text{ to } h$$

the weight update law is

$$w_i(t+1) = w_i(t) - \eta_2 \frac{\partial E}{\partial w_i}$$

where the cost function is $E = \frac{1}{2} \sum (y^d - y)^2$

Contd...

The actual response is

$$y = \sum_{i=1}^{h} \phi_i w_i$$

the activation function is taken as

$$\phi_i = e^{-z_i^2/2\sigma^2}$$

where $z_i = \|x - c_i\|$, σ is the width of the center. Differentiating E w.r.t. w_i, we get

$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial y} \times \frac{\partial y}{\partial w_i} = -(y^d - y)\phi_i$$

Contd...

Differentiating E w.r.t. c_{ij}, we get

$$\begin{aligned}\frac{\partial E}{\partial c_{ij}} &= \frac{\partial E}{\partial y} \times \frac{\partial y}{\partial \phi_i} \times \frac{\partial \phi_i}{\partial c_{ij}} \\ &= -(y^d - y) \times w_i \times \frac{\partial \phi_i}{\partial z_i} \times \frac{\partial z_i}{\partial c_{ij}}\end{aligned}$$

$$\text{Now,} \quad \frac{\partial \phi_i}{\partial z_i} = -\frac{z_i}{\sigma^2}\phi_i$$

$$\begin{aligned}\text{and} \quad \frac{\partial z_i}{\partial c_{ij}} &= \frac{\partial}{\partial c_{ij}}\Big(\sum_j (x_j - c_{ij})^2\Big)^{1/2} \\ &= -(x_j - c_{ij})/z_i\end{aligned}$$

Contd...

After simplification, the update rule for center learning is:

$$c_{ij}(t+1) \;=\; c_{ij}(t) + \eta_1 (y^d - y) w_i \frac{\phi_i}{\sigma^2}(x_j - c_{ij})$$

The update rule for the linear weights is:

$$w_i(t+1) = w_i(t) + \eta_2 (y^d - y)\phi_i$$

Example: System identification

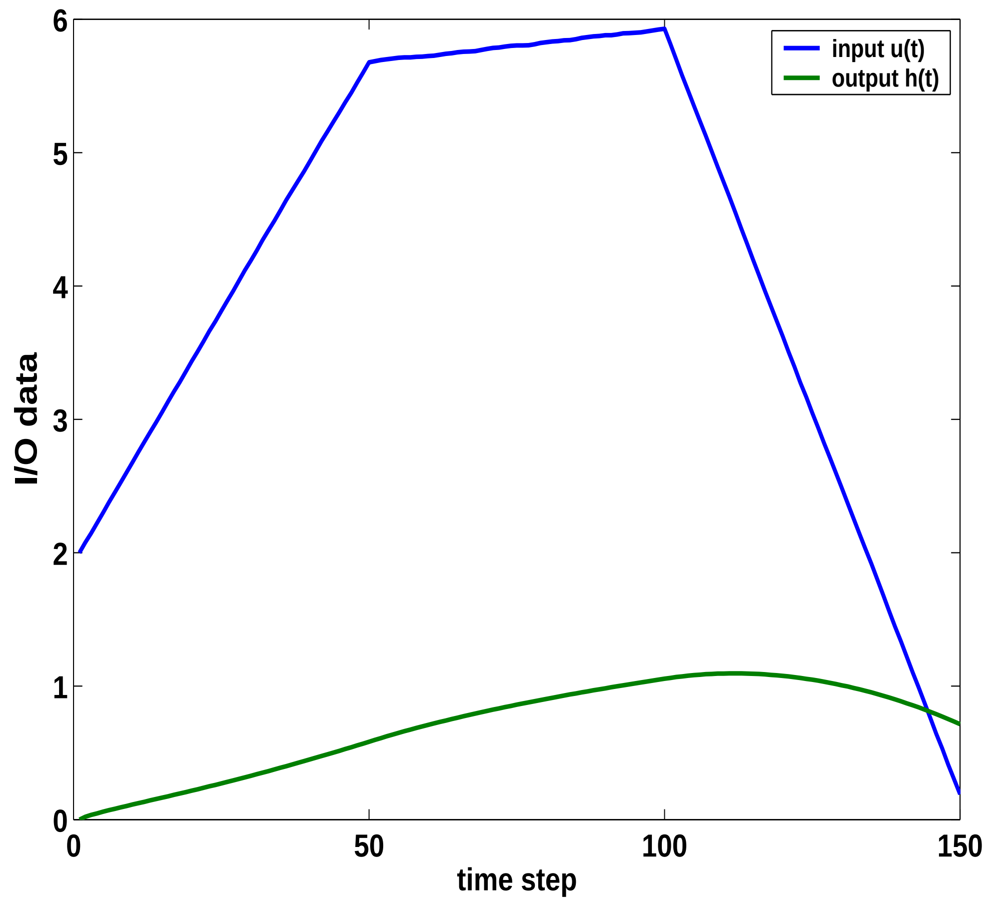
The same Surge Tank system has been taken for simulation. The system model is given as

$$h(t+1) = h(t) + T\left(\frac{-\sqrt{2gh(t)}}{\sqrt{3h(t)+1}} + \frac{u(t)}{\sqrt{3h(t)+1}}\right)$$

t	:	discrete time step
T	:	sampling time
$u(t)$	:	input flow, can be positive or negative
$h(t)$	:	liquid level of the tank (output)
g	:	the gravitational acceleration

Data generation

Sampling time is taken as 0.01 sec, 150 data have been generated using the system equation. The nature of input $u(t)$ and $y(t)$ is shown in the following figure.



Contd...

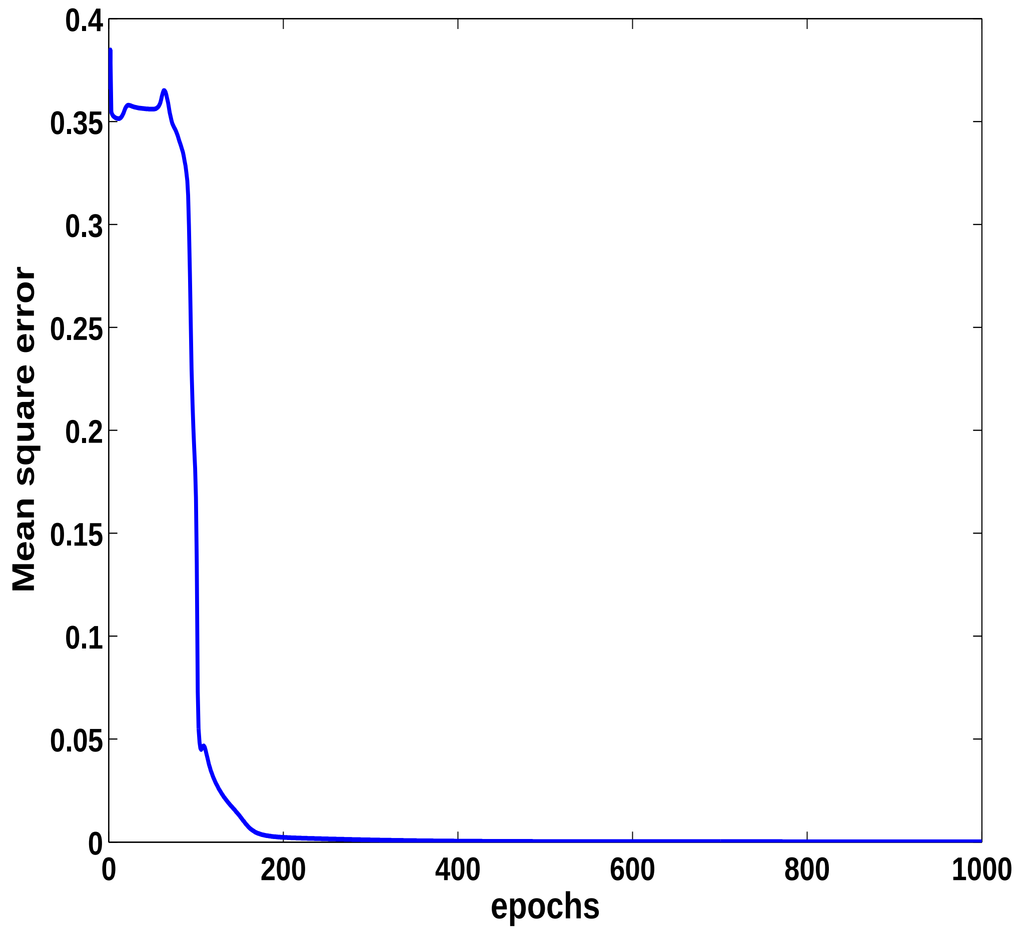
The system is identifi ed from the input-output data using a radial basis function network

Network parameters are given below:

Number of inputs	:	$2\ [(u(t), h(t)]$
Number of outputs	:	1 [target: $h(t+1)$]
Units in the hidden layer	:	30
Number of I/O data	:	150
Radial Basis Function	:	Gaussian
Width of the radial function (σ)	:	0.707
Center learning rate (η_1)	:	0.3
Weight learning rate (η_2)	:	0.2

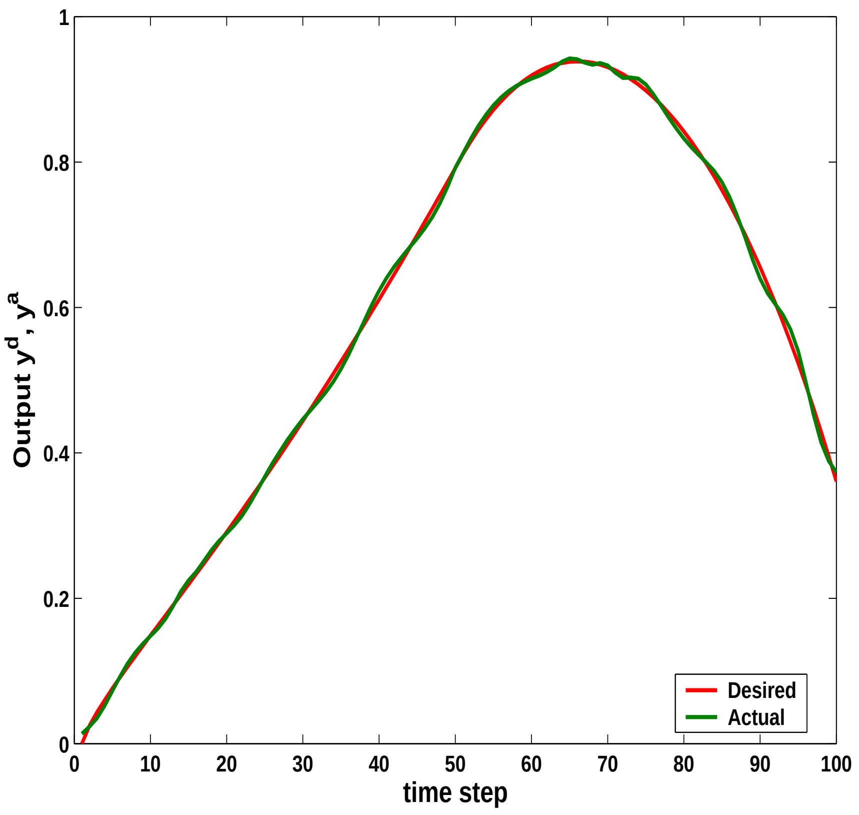
Result: Identification

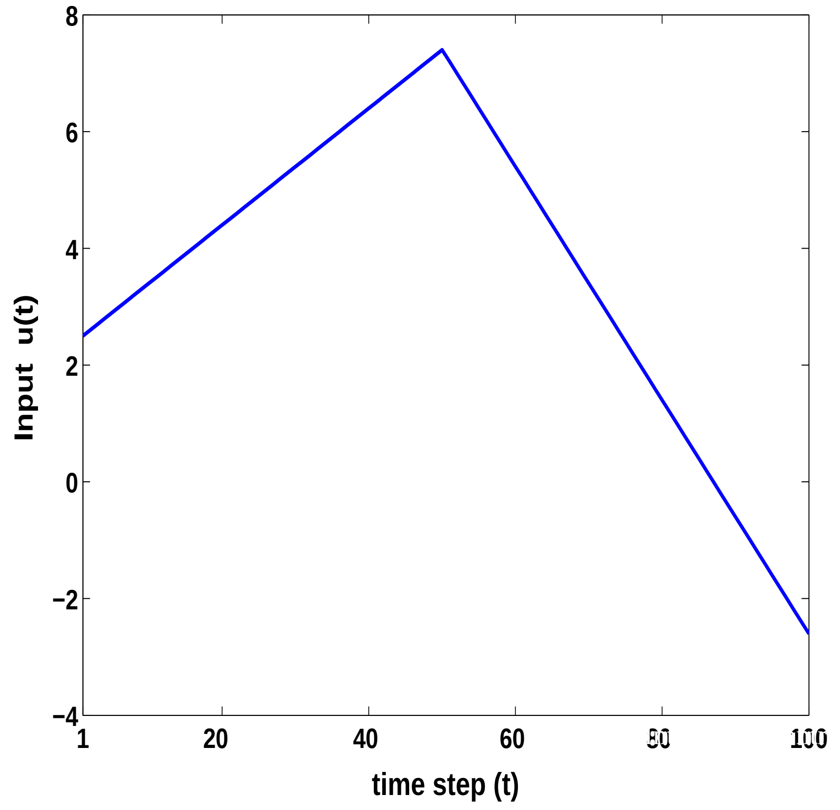
After identification, the root mean square error error is found to be < 0.007. Convergence of mean square error is shown in the following figure.



Model Validation

After identification, the model is validated through a set of 100 input-output data which is different from the data set used for training. The result is shown in the following figure where left figure represents the desired and network output and right figure shows the corresponding input.





Hybrid Learning

In hybrid learning, the radial basis functions relocate their centers in a self-organized manner while the weights are updated using supervised learning.

When a pattern is presented to RBFN, either a new center is grown if the pattern is sufficiently novel or the parameters in both layers are updated using gradient descent.

The test of novelty depends on two criteria:

- Is the Euclidean distance between the input pattern and the nearest center greater than a threshold $\delta(t)$?
- Is the mean square error at the output greater than a desired accuracy?

A new center is allocated when both criteria are satisfied.

Contd...

Find out a center that is closest to x in terms of Euclidean distance. This particular center is updated as follows:

$$c_i(t+1) = c_i(t) + \alpha(x - c_i(t))$$

Thus the center moves closer to x.

While centers are updated using unsupervised learning, the weights can be updated using least mean squares (LMS) or recursive least squares (RLS) algorithm. We will present the RLS algorithm.

Contd...

The i^{th} input of the RBFN can be written as

$$\hat{x}_i = \phi^T \theta_i, \quad i = 1, \cdots, n$$

where $\phi \in R^l$, the output vector of the hidden layer; $\theta_i \in R^l$, the connection weight vector from hidden units to i^{th} output unit. The weight update law:

$$\hat{\theta}_i(t+1) = \hat{\theta}_i(t) + P(t+1)\phi(t)[x_i(t+1) - \phi^T(t)\hat{\theta}_i(t)$$

$$P(t+1) = P(t) - P(t)\phi(t)[1 + \phi^T(t)P(t)\phi(t)]^{-1}\phi^T(t)P(t)$$

where $P(t) \in R^{l \times l}$. This algorithm is more accurate and fast compared to LMS algorithm.

Example: System Identification

The same surge tank model is identifi ed with the same input-output data set. The parameters of RBFN are also same as that of gradient descent technique.

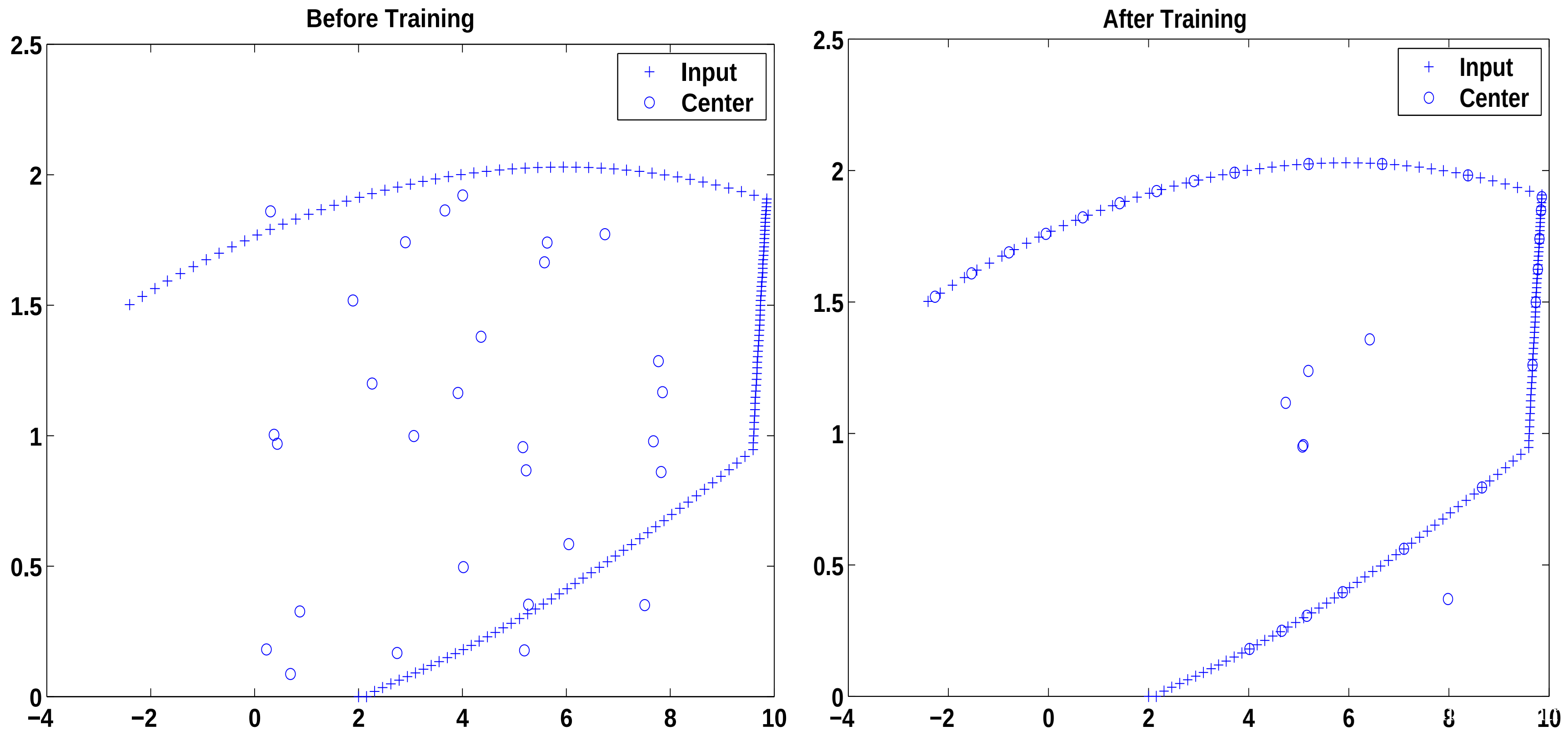
Center training is done using unsupervised K-mean clustering with a learning rate of 0.5.

Weight training is done using gradient descent method with a learning rate of 0.1.

After identifi cation root mean square is found to be < 0.007.

Contd...

Mapping of input data and centers are shown in the following figure. Left figure shows how the centers are initialized and right figure shows the spreading of centers towards the nearest inputs after learning.



Comparison of Results

Schemes	RMS error	No. of iterations
Back-propagation	0.00822	~ 5000
RBFN (Gradient Descent)	0.00825	~ 2000
RBFN (Hybrid Learning)	0.00823	~ 1000

It is seen from the table that RBFN learns faster compared to a simple feedforward network.