CHAPTER IX

Radial Basis Function Networks

Radial basis function (RBF) networks are feed-forward networks trained using a supervised training algorithm. They are typically configured with a single hidden layer of units whose activation function is selected from a class of functions called basis functions. While similar to back propagation in many respects, radial basis function networks have several advantages. They usually train much faster than back propagation networks. They are less susceptible to problems with non-stationary inputs because of the behavior of the radial basis function hidden units.

Popularized by Moody and Darken (1989), RBF networks have proven to be a useful neural network architecture. The major difference between RBF networks and back propagation networks (that is, multi layer perceptron trained by Back Propagation algorithm) is the behavior of the single hidden layer. Rather than using the sigmoidal or S-shaped activation function as in back propagation, the hidden units in RBF networks use a Gaussian or some other basis kernel function. Each hidden unit acts as a locally tuned processor that computes a score for the match between the input vector and its connection weights or centers. In effect, the basis units are highly specialized pattern detectors. The weights connecting the basis units to the outputs are used to take linear combinations of the hidden units to product the final classification or output.

In this chapter first the structure of the network will be introduced and it will be explained how it can be used for function approximation and data interpolation. Then it will be explained how it can be trained.

9.1 The Structure of the RBF Networks

Radial Basis Functions are first introduced in the solution of the real multivariable interpolation problems. Broomhead and Lowe (1988), and Moody and Darken (1989) were the first to exploit the use of radial basis functions in the design of neural networks.

The structure of an RBF networks in its most basic form involves three entirely different layers (Figure 9.1).

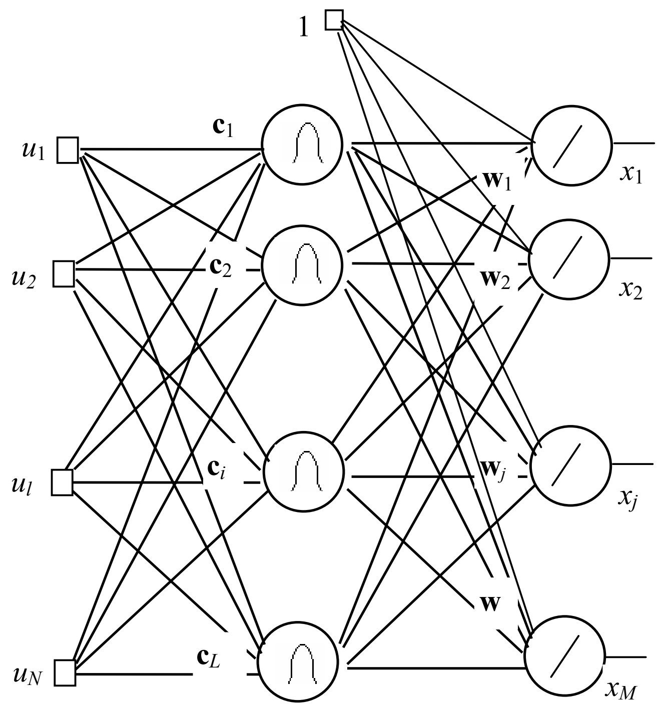

Fig. 9.1. Structure of the Standart RBF network

The input layer is made up of source nodes (sensory units) whose number is equal to the dimension p of the input vector **u**.

9.1.1 Hidden layer

The second layer is the hidden layer which is composed of nonlinear units that are connected directly to all of the nodes in the input layer. It is of high enough dimension, which serves a different purpose from that in a multilayer perceptron.

Each hidden unit takes its input from all the nodes at the components at the input layer. As mentioned above the hidden units contains a basis function, which has the parameters center and width. The center of the basis function for a node *i* at the hidden layer is a vector $\mathbf{c}_i$ whose size is the as the input vector **u** and there is normally a different center for each unit in the network.

First, the radial distance d_i, between the input vector **u** and the center of the basis function $\mathbf{c}^i$ is computed for each unit *i* in the hidden layer as

$$d_i = \|\mathbf{u} - \mathbf{c}_i\| \tag{9.1.1}$$

using the Euclidean distance.

The output h_i of each hidden unit *i* is then computed by applying the basis function G to this distance

$$h_i = \mathrm{G}(d_i, \sigma_i) \tag{9.1.2}$$

As it is shown in Figure 9.2, the basis function is a curve (typically a Gaussian function, the width corresponding to the variance, σ_i) which has a peak at zero distance and it decreases as the distance from the center increases.

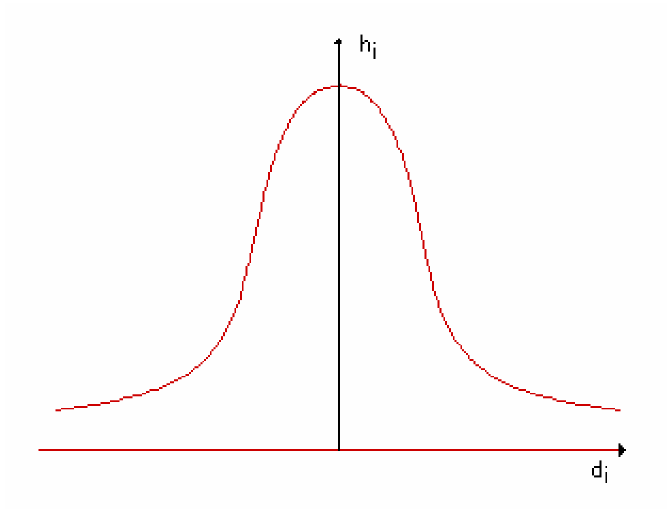

Figure 9.2. The response region of an RBF hidden node around its center as a function of the distance from this center.

For an input space $\mathbf{u}\in R^2$, that is M=2, this corresponds to the two dimensional Gaussian centered at $\mathbf{c}_i$ on the input space, where also $\mathbf{c}_i \in R^2$, as it is shown in Figure 9.3

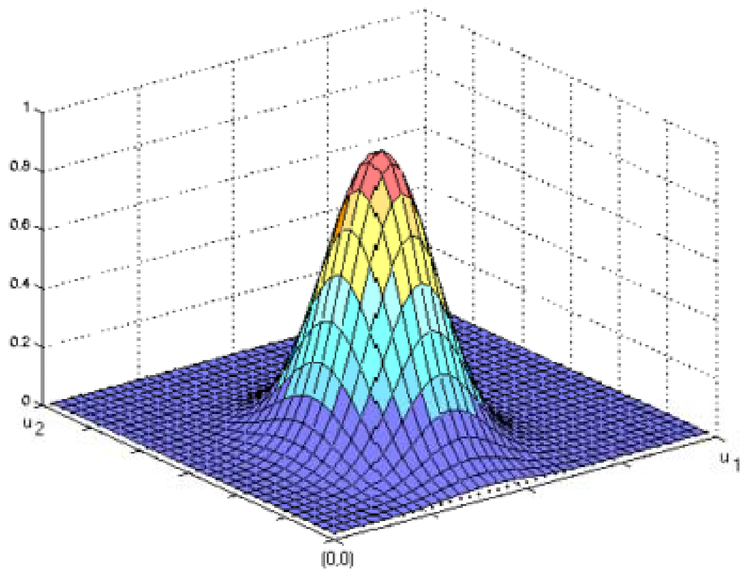

Figure 9.3 Response of a hidden unit on the input space for $u\in R^2$

9.1.2 Output layer

The transformation from the input space to the hidden unit space is nonlinear, whereas the transformation to the hidden unit space to the output space is linear.

The j^th output is computed as

$$x_j = \mathrm{f}_j(\mathbf{u}) = w_{0j} + \sum_{i=1}^{L} w_{ij} h_i \quad j = 1,2,..,M \tag{9.1.3}$$

9.1.3 Mathematical model

In summary, the mathematical model of the RBF network can be expressed as:

$\mathbf{x}=\mathbf{f}(\mathbf{u})$, $\mathbf{f}:\mathrm{R}^N \rightarrow \mathrm{R}^M$

$$x_j = \mathrm{f}_j(\mathbf{u}) = w_{0j} + \sum_{i=1}^{L} w_{ij} G(\|\mathbf{u} - \mathbf{c}_i\|) \quad j = 1,2,..,M \tag{9.1.5}$$

where is the the Euclidean distance between $\mathbf{u}$ and $\mathbf{c}_i$

9.2 Function approximation

Let y=g(u) be a given function of u, $y \in \mathrm{R}$, $u \in \mathrm{R}$, $g:\mathrm{R} \rightarrow \mathrm{R}$, and let G_i i=1..L, be a finite set of basis functions.

The function g can be written in terms of the given basis functions as

$$y = g(\mathbf{u}) = \sum_{i=1}^{L} w_i G_i(u) + \mathrm{r}(u) \tag{9.2.1}$$

where r(u) is the residual.

The function y can be approximated as

$$y = g(\mathbf{u}) \cong \sum_{i=1}^{L} w_i G_i(u) \qquad (9.2.2)$$

The aim is to minimize the error by setting the parameters of G$_i$ appropriately. A possible choice for the error definition is the L2 norm of the residual function r(u) which is defined as

$$\|r(u)\|_{L2} = \int r(u)^2 \qquad (9.2.3)$$

9.2.1 Approximation by RBFNN

Now, consider the single input single output RBF network shown in Figure 9.4. Then *x* can be written as

$$x = \mathrm{f}(\mathbf{u}) = \sum_{i=1}^{L} w_i G_i(\|\mathbf{u} - \mathbf{c}_i\|) \qquad (9.2.4)$$

By the use of such a network, y can be written as

$$y = \sum_{i=1}^{L} w_i G(\|u - c_i\|) + \mathrm{r}(u) = \mathrm{f}(u) + \mathrm{r}(u) \qquad (9.2.5)$$

where f(u) is the output of the RBFNN given in Figure 9.4 and r(u) is the residual. By setting the center c$_{i,}$ the variance σ_i , and the weight w_i the error appropriately, the error can be minimized.

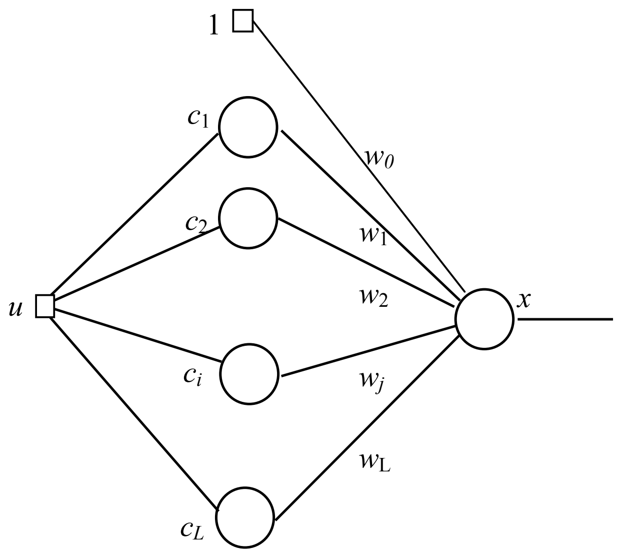

Figure 9.4 Single input, single output RBF network

Whatever we discussed here for g:R→R, can be generalized to **g**:R^N→R^M easily by using an *N* input, **M** output RBFNN given in figure 9.1 previously.

9.2.2 Data Interpolation

Given input output training patterns ($\mathbf{u}^k$,$\mathbf{y}^k$), *k*=1,2, ..*K,* the aim of data interpolation is to approximate the function **y** from which the data is generated. Since the function **y** is unknown, the problem can be stated as a minimization problem which takes only the sample points into consideration:

Choose $w_{i,j}$ and $\mathbf{c}_i$, *i*=1,2...*L, j*=1,2...*M* so as to minimize

$$J(w,c)=\sum_{k=1}^{K}\left\|y^k-f(u^k)\right\|^2 \tag{9.2.6}$$

As an example, the output of an RBF network trained to fit the data points given in Table 9.1 is given in Figure 9.5.

TABLE I: 13 data points generated by using sum of three gaussians with c_1=0.2000 c_2=0.6000 c_3=0.9000 w_1=0.2000 w_2=0.5000 w_3=0.3000 σ=0.1000

data no	1	2	3	4	5	6	7	9	10	11	12	13
x	0.0500	0.2000	0.2500	0.3000	0.4000	0.4300	0.4800	0.6000	0.7000	0.8000	0.9000	0.9500
f(x)	0.0863	0.2662	0.2362	0.1687	0.1260	0.1756	0.3290	0.6694	0.4573	0.3320	0.4063	0.3535

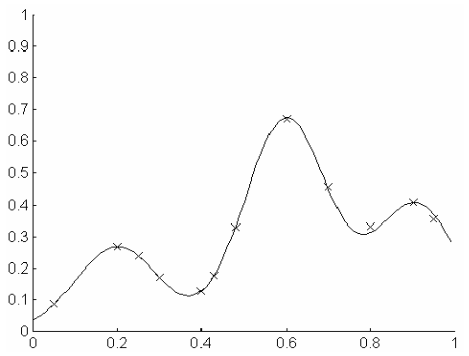

Figure 9.5 Output of the RBF network trained to fit the datapoints given in Table 9.1

9.3 Training RBF Networks

The training of a RBF network can be formulated as a nonlinear unconstrained optimization problem given below:

Given input output training patterns ($\mathbf{u}^k$,$\mathbf{y}^k$), k=1,2, ..K, choose $w_{i,j}$ and $\mathbf{c}_i$,, i=1,2...L, j=1,2...M so as to minimize

$$J(w,c)=\sum_{k=1}^{K}\left\|y^k - f(u^k)\right\|^2 \qquad (9.3.1)$$

(1.3)

Note that the training problem becomes quadratic once if $\mathbf{c}_i$'s (radial basis function centers) are known.

9.3.1 Adjusting the widths

In its simplest form, all hidden units in the RBF network have the same width or degree of sensitivity to inputs. However, in portions of the input space where there are few patterns, it is sometime desirable to have hidden units with a wide area of reception. Likewise, in portions of the input space, which are crowded, it might be desirable to have very highly tuned processors with narrow reception fields. Computing these individual widths increases the performance of the RBF network at the expense of a more complicated training process.

9.3.2 Adjusting the centers

Remember that in a back propagation network, all weights in all of the layers are adjusted at the same time. In radial basis function networks, however, the weights into the hidden layer basis units are usually set before the second layer of weights is adjusted. As the input moves away from the connection weights, the activation value falls off. This behavior leads to the use of the term “center” for the first-layer weights. These center weights can be computed using Kohonen feature maps, statistical methods such as K-Means clustering, or some other means. In any case, they are then used to set the areas of sensitivity for the RBF network’s hidden units, which then remain fixed.

9.3.3 Adjusting the weights

Once the hidden layer weights are set, a second phase of training is used to adjust the output weights. This process typically uses the standard steepest descent algorithm. Note that the training problem becomes quadratic once if $\mathbf{c}_i$'s (radial basis function centers) are known.