UNIT 8 RADIAL BASIS FUNCTION NETWORKS

Structure **Page No.**

8.1 INTRODUCTION

We now move on to discuss a class of feed forward neural networks called radial basis function networks (RBFN) that compute activations at the hidden neurons in a way that is different from what we have seen in the case of feed forward neural networks. RBFNs are computed using an exponential of a distance measure between the input vector and a prototype vector that characterizes the signal function at a hidden neuron. We shall discuss RBFN in Sec. 8.2. We shall continue calculations of these networks for Backward and forward propagation in Sec. 8.3. At last in Sec. 8.4 we shall discuss some application of there networks.

Objectives

After studying this unit you should be able to:

- define the radial basis function networks;
- calculate the modified weight using backward and forward propagation;
- define the applications.

8.2 RADIAL BASIS FUNCTION NETWORK (RBFN)

A radial basis function network is a neural network approached by viewing the design as a curve-fitting (approximation) problem in a high dimensional space. In a neural network, the hidden units form a set of "functions" that compose a random "basis" for the input patterns (vectors). These functions are called radial basis functions.

A radial basis function (RBF) is a special type of function, whose response decreases or increases monotonically with its distance from a central point. Various types of RBFs are given below:

1. **Thin plate spline function:** $f(x) = x^2 \log(x)$,
2. **Gaussian function:** $f(x) = \exp\left(-\frac{\|x-\mu\|^2}{2\sigma^2}\right)$, where μ and σ indicate the mean and standard deviation of the distribution, respectively,
3. **Multi-quadratic function:** $f(x) = \sqrt{x^2 + \sigma^2}$,
4. **Inverse multi-quadratic function:** $f(x) = \frac{1}{\sqrt{x^2 + \sigma^2}}$.

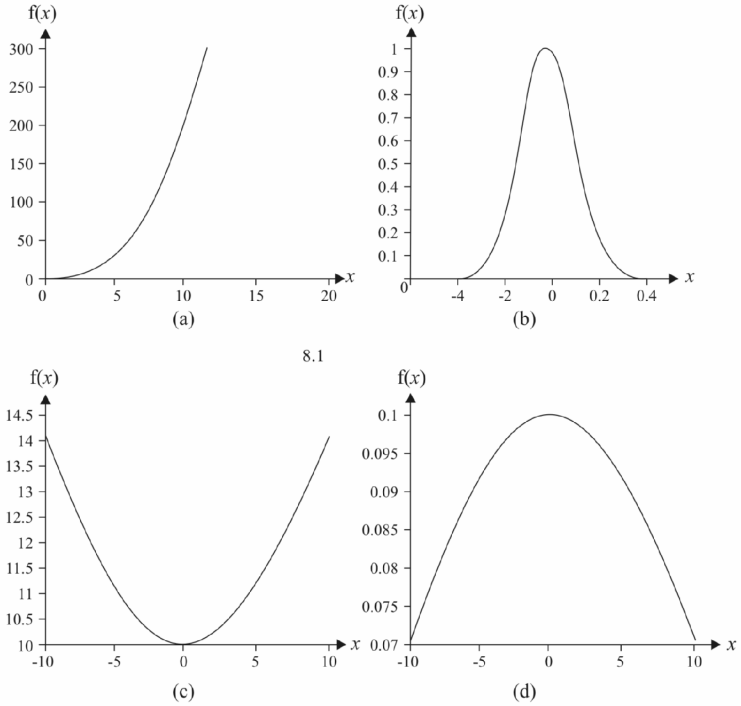

Fig. 1: Graphs various radial basis function

Fig. 1 shows the plots of the above RBFs, (a): Plate spline function, (b): Gaussian function, (c): Multi-quadratic function and (d): Inverse multi-quadratic function.

In 1988, Broomhead and Lowe proposed Radial Basis Function Network (RBFN). It is a special type of Artificial Neural Network (ANN) that can fit a surface in multi-dimensional space after ensuring the best match to the training data. These networks can tackle function approximation, the problems related to time-series forecasting, classification, clustering, recognition of handwritten characters, face recognition, voice identification and others. This network consists of an input layer, hidden layer and output layer. The input layer consists of some input data nodes. The number of these nodes is equal to the number of independent input variables of the system to be modeled. The hidden layer of a few neurons is with radial basis transfer functions. The output layer composed of some neurons (number is made equal to the number with linear transfer function usually, and the number of outputs}. The performance of the RBFN is dependent on the number of hidden neurons, their centers and radii. For example, the performance of a network with Gaussian transfer functions depends on their centers and radii, which are nothing but their means and standard deviations, respectively. The network with an insufficient number of hidden neurons may not be able to offer good approximation. On the other hand, a large number of hidden neurons may yield good approximation but at the cost of predictive power. It is known as an **over-fitting problem** of the network. The RBFN with either narrow or wide radii may produce some bad results. Thus, a proper tuning of the network is necessary to carry out.

RBFN are usually trained by the following methods:

i) The number of hidden neurons is pre-defined and suitable values of their centers and radii are determined through a proper training.

ii) The algorithm starts with an empty set of hidden neurons, which grow in number until a given condition is reached. Thus, the number of hidden neurons and their centers and radii are searched iteratively.

iii) The algorithm begins with a large number of hidden neurons. Some of the hidden neurons and their connecting weights are removed iteratively and

ultimately, the optimal network is obtained.

The input-output relationship of a RBFN with N independent input variables and M output is shown in Fig. 2.

Let us suppose that the network consists of an input layer having N input nodes, one hidden layer containing P neurons and an output layer of M neurons. Let us also assume that a set of L training scenarios will be used to train the network. For a particular training scenario (say-l-th), the forward and backward propagation are discussed in the following section.

8.3 FORWARD AND BACKWARD PROPAGATION

The following steps are considered in the forward calculations:

i) The outputs of input layer

Let us represent L training scenarios, which constitute the input vector, as given

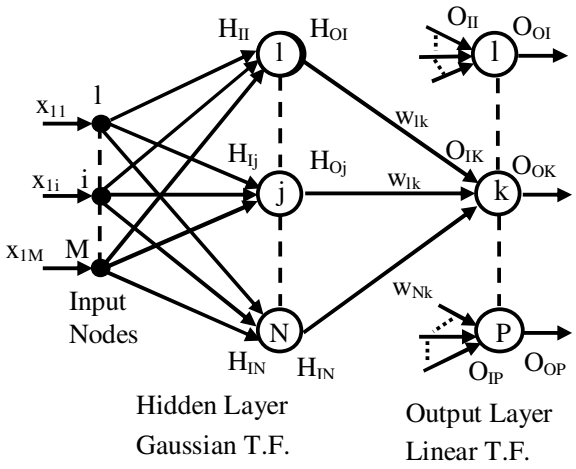

Fig. 2: Radial Basis Function Network.

below:

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_l \\ \vdots \\ x_L \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1i} & \cdots & x_{1N} \\ x_{21} & x_{22} & \cdots & x_{2i} & \cdots & x_{2N} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{l1} & x_{12} & \cdots & x_{li} & \cdots & x_{lN} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{L1} & x_{L2} & \cdots & x_{Li} & \cdots & x_{LN} \end{bmatrix}$$

Let us assume that 1-th training scenario (that is, $x_{l1}, x_{l2,} \ldots, x_{li,} \ldots, x_{lM}$ expressed in the normalized form) is passed through the network. The outputs of different nodes of the input layer are nothing but the inputs of corresponding nodes.

ii) The outputs of hidden layer

The hidden layer consists of p neurons. Each of them has a Gaussian transfer function represented by a separate set of mean μ and standard deviation σ values. Thus, the values of μ and σ associated with the transfer functions of 1-st, 2-nd, …, p-th neurons are represented by $(\mu_1, \sigma_1), (\mu_1, \sigma_2), \cdots, (\mu_p, \sigma_p)$,

respectively. The output of j-th hidden neuron can be expressed like the following:

$$[O_H]_j = \exp\left[-\frac{\|x_1 - \mu_j\|^2}{2\sigma_j^2}\right] \quad (1)$$

iii) The inputs of output layer

The input of k-th neuron lying on output layer can be represented as follows:

$$[I_O]_k = \sum_{j=1}^{N} w_{Jk}\, O_{Hj} \quad (2)$$

where $w_{jk`}$ indicates the connecting weight between j-th hidden neuron and k-th output neuron.

iv) The output of output layer

The output of k-th neuron lying on output layer is determined as follows:

$$[O_O]_k = [I_O]_k \quad (3)$$

here the output neurons are assumed to have linear transfer function.

Let us suppose that the target output of k-th output neuron is denoted by T_{Ok}. Therefore, the error in prediction can be calculated like the following:

$$E_k = \frac{1}{2}(T_{Ok} - O_{Ok})^2. \quad (4)$$

Back-propagation algorithm can be implemented through either an incremental or a batch mode of training as discussed below.

Let us consider an incremental mode of training, that is, the connecting weight, mean and standard deviation values of the Gaussian distribution will be updated to minimize the error in prediction, corresponding to l-th training scenario.

i) Weight updating

The weights are updated following the rule given below.

$$w_{t+1} = w_t + \Delta w, \quad (5)$$

where Δw indicates the change in w value. The change in w value at t-th iteration is given by the following expression:

$$\Delta w_{jk}(t) = -\eta \frac{\partial E_k}{\partial w_{jk}}(t) + \alpha' \Delta w_{jk}(t-1), \quad (6)$$

where η and α' represent the learning rate and momentum constant, respectively. Now, $\frac{\partial E_k}{\partial w_{jk}}$ can be determined corresponding to k-th output neuron, using the chain rule of differentiation as given below.

$$\frac{\partial E_k}{\partial w_{jk}} = \frac{\partial E_k}{\partial O_{Ok}} \frac{\partial O_{Ok}}{I_{Ok}} \frac{\partial I_{Ok}}{\partial w_{jk}}, \tag{7}$$

where $\frac{\partial E_k}{\partial O_{Ok}} = -(T_{Ok} - O_{Ok})$,

$\frac{\partial O_{Ok}}{\partial O_{Ik}} = 1$,

$\frac{\partial O_{Ik}}{\partial w_{jk}} = O_{Hj}$

Therefore, Eq. (7) can be re-written as

$$\frac{\partial E_k}{\partial w_{jk}} = -(T_{Ok} - O_{Ok}) O_{Hj} \tag{8}$$

Now, Eq. (5) can be expressed like the following:

$$w_{jk,\,updated} = w_{jk,\,previous} + \eta(T_{Ok} - O_{Ok}) H_{Oj} + \alpha' \Delta w_{jk,\,previous.} \tag{9}$$

The similar procedure can be followed to determine updated values of other weights.

ii) Mean updating

The mean of Gaussian distribution corresponding to j-th hidden neuron can be updated as follows:

$$\mu_{j,\,updated} = \mu_{j,\,previous} + \Delta\mu_j, \tag{10}$$

where the change in μ at t-th iteration can be determined like the following:

$$\Delta\mu_j(t) = -\eta \left\{ \frac{\partial E}{\partial \mu_j}(t) \right\} av + \alpha' \Delta\mu_j(t-1), \tag{11}$$

where $\left\{ \frac{\partial E}{\partial \mu_j} \right\} av = \frac{1}{p} \sum_{k=1}^{P} \frac{\partial E_k}{\partial \mu_j}$

$$\frac{\partial E_k}{\partial \mu_j} = \frac{\partial E_k}{\partial O_{Ok}} \frac{\partial O_{Ok}}{\partial I_{Ok}} \frac{\partial I_{Ok}}{\partial O_{Hj}} \frac{\partial O_{Hj}}{\partial \mu_j} \tag{12}$$

We have already seen that $\frac{\partial E_k}{\partial O_{Ok}} \frac{\partial O_{Ok}}{\partial I_{Ok}} = -(T_{Ok} - O_{Ok})$.

Now, $\frac{\partial I_{Ok}}{\partial O_{Hj}} = w_{jk,}$

$$\frac{\partial O_{Hj}}{\partial \mu_j} = O_{Hj} \left\{ \frac{(x_{l1} + x_{12} + \cdots + x_{1i} + \cdots + x_{lM}) - M\mu_j}{\sigma_j^2} \right\}.$$

Therefore, we get

$$\frac{\partial E_k}{\partial \mu_j} = -(T_{Ok} - O_{Ok}) w_{jk} O_{Hj} \left\{ \frac{(x_{l1} + x_{l2} + \cdots + x_{li} + \cdots + x_{lM}) - M\mu_j}{\sigma_j^2} \right\}. \quad (13)$$

iii) Standard Deviation Updating

The standard deviation corresponding to j-th hidden neuron will be updated as follows:

$$\sigma_{j,\text{updated}} = \sigma_{j,\text{previous}} + \Delta\sigma_j, \quad (14)$$

where the change in σ at i-th iteration is written like the following:

$$\Delta\sigma_j(t) = -\eta \left\{ \frac{\partial E}{\partial \sigma_j}(t) \right\} av + \alpha' \Delta\sigma_j(t-1), \quad (15)$$

where $\left\{ \frac{\partial E}{\partial \sigma_j} \right\} av = \frac{1}{P} \sum_{k=1}^{P} \frac{\partial E_k}{\partial \sigma_j}$,

$$\frac{\partial E_k}{\partial \sigma_j} = \frac{\partial E_k}{\partial O_{Ok}} \frac{\partial O_{Ok}}{\partial I_{Ok}} \frac{\partial I_{Ok}}{\partial O_{Hj}} \frac{\partial O_{Hj}}{\partial \sigma_j}. \quad (16)$$

We have already seen that $\frac{\partial E_k}{\partial O_{Ok}} \frac{\partial O_{Ok}}{\partial I_{Ok}} = -(t_{Ok} - O_{Ok})$.

Now, $\frac{\partial I_{Ok}}{\partial O_{Hj}} = w_{jk}$,

$$\frac{\partial O_{Hj}}{\partial \sigma_j} = O_{Hj} \left\{ \frac{(x_{l1} - \mu_j)^2 + (x_{l2} - \mu_j)^2 + \cdots + (x_{li} - \mu_j)^2 + \cdots + (x_{lM} - \mu_j)^2}{\sigma_j^3} \right\}.$$

Therefore, we get

$$\frac{\partial E_k}{\partial \sigma_j} = -(T_{Ok} - O_{Ok}) w_{jk} O_{Hj} \left\{ \frac{\sum_{i=1}^{M} (x_{li} - \mu_j)^2}{\sigma_j^3} \right\}. \quad (17)$$

8.4 APPLICATIONS

RBNF can be applied for the following

i) Polynomial Fitting

For comparison purpose, Fig. 3 shows the results obtained by the following four methods; linear interpolation, cubic spline interpolation, fourth-order polynomial interpolation, and third-order polynomial approximation. All the methods except for the first one can generate smooth curves. However, it is cumbersome to extend spline and polynomial interpolation to high-dimensional data sets that can be handled by the interpolation RBFN easily.

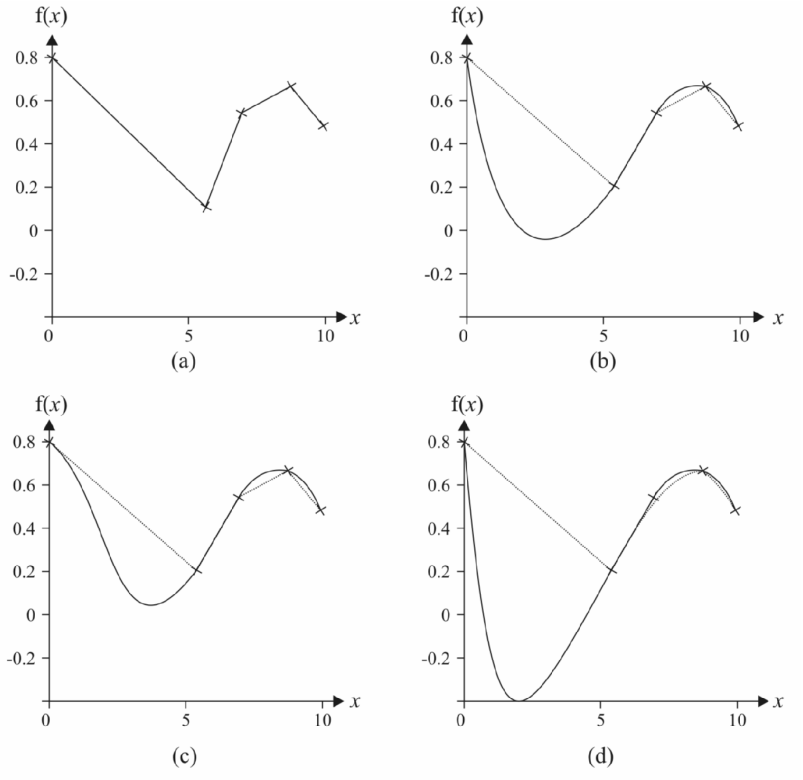

Fig. 3: Interpolation and approximation results obtained by four polynomial interpolation/approximation methods, (a): Linear interpolation, (b): Cubic spine interpolation, (c): Fourth-order polynomial interpolation and (d): Third-order polynomial approximation

ii) Backpropagation MLP Fitting

Fig. 4 presents the results obtained by a backpropagation MLP. In the MLP results, six choices for the number of hidden units are considered to see how the overall responses are affected. The curves generated by MLP are quite smooth and they all pass the five given data points. However, the positions of these curves between data points are strongly influenced by the initial weights of the MLP. Moreover, as more hidden nodes are used, the variations of curves between data points are more pronounced.

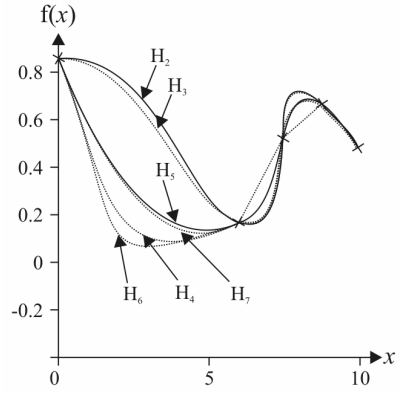

Fig. 4: Interpolation results obtained by simple backpropagation MLPs. "#HU" denoted "number of hidden units"

Now, let us summarize the unit.

8.5 SUMMARY

In this unit we have covered the following points.

1. A radial basis function network is a neural network approached by viewing the design as a curve-fitting (approximation) problem in a high dimensional space.
2. In forward propagation, the outputs of input layer, the output of the hidden layer, the inputs of output layer and the outputs of output layer were determined.
3. In backward propagation, the weights, mean, standard deviations were updated.
4. The application of RBFN.