Chapter 2

Review of Radial Basis functions and Bootstrap Method

2.1 Neural Network

Any human response comes from the nervous system. A stimulus enters the system by the receptors and is recognized by the system. Then, the recognition is represented by the human effectors. Figure 2.1 shows the human nervous cells (neuron) which are an element of the nervous system. A dendrite zone is an entrance of any input. Recognition or decision is decided by the cell body. The axon line spends the result of the decision out to the other neuron next. The system consists of a large number of these neurons.

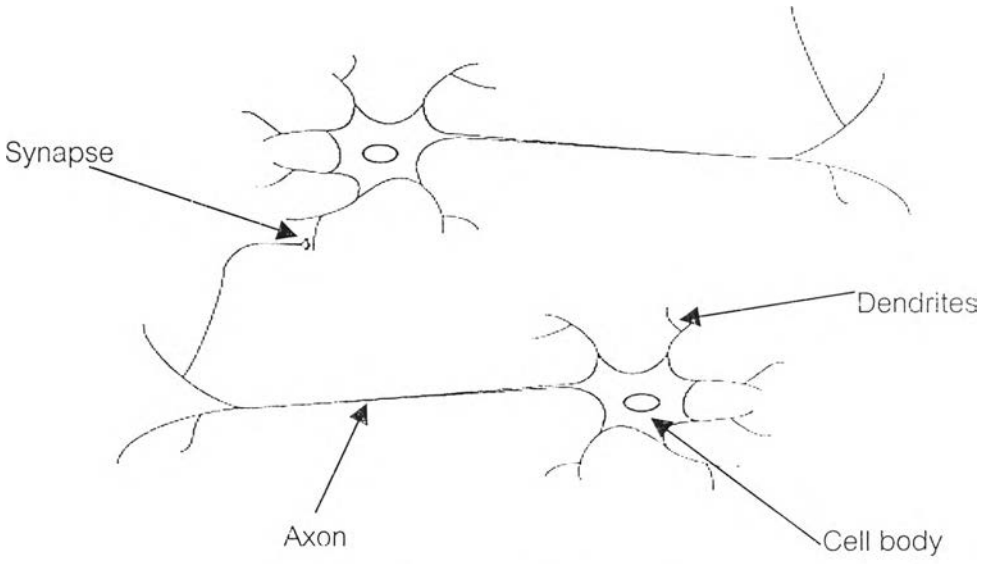

Figure 2.1 Drawing of Biological Neurons.

Each neuron in the system may be viewed as an information processor. The flow of information in each neuron begins at the receptive zone (dendrites). The cell body summarizes them effectively and thresholds them to create the result. Then, the result is spent out pass through the transmission line (axon). Between neurons, there are synapses that weight the information of each transmitter. The arrangements of neurons create the cooperative neural network system.

Normally, the learning process adjusts the synaptic weights to store the acquired knowledge [1]. A neural network is a machine that models a task or a function by using electronic components or simulated software. In the modeling, the interconnection strengths (synaptic weights) are used to store the knowledge. The weight adjusting is called learning process. A model of information-processing node (neuron) in term of computational is show in Figure 2.2.

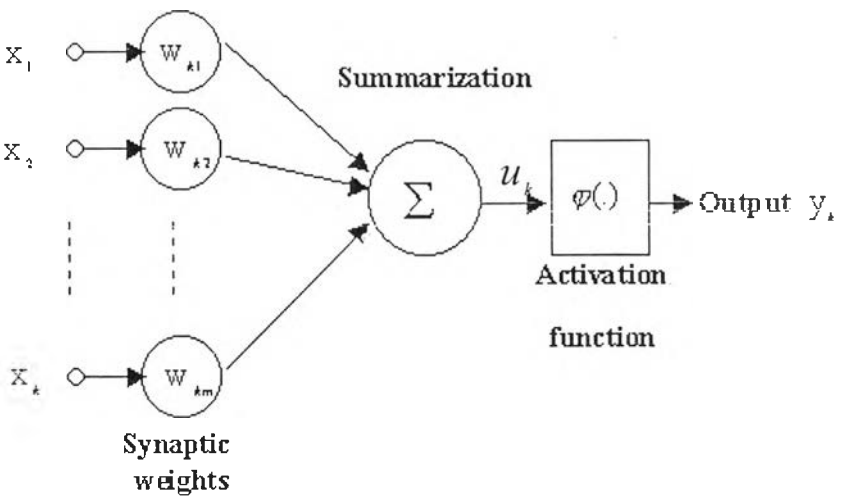

Figure 2.2 A model of a neuron.

Figure 2.2 shows the model of a neuron (k). When the neuron receives the input signals of m elements, the neuron weights them with the synaptic weights and, then, summarizes them to an input u_k of the activation function φ. Then, the activation function activates the output of this neuron as y_k.

The model can be described in a mathematical term with the following equations.

$$u_k = \sum_{j=0}^{m} w_{kj} x_j \tag{2.1}$$

$$y_k = \varphi(u_k) \tag{2.2}$$

where k is the index of the neuron, j is the index of the input vectors, x_j is the input vectors, w_{kj} is the weight vectors, and y_k is the output. The activation function defines the output of a neuron. So the modeling of the neural networks have to choose a kind of an activation function [1]. We are interested in a kind of neural network model that activates correctly with a radial basis function, which is used to perform as the recognizer of the neuron (node), that is described in the next section.

We concentrate on a two dimensional space of classification problem, and define the meaning of the following words used in this thesis as follows:

- Training data is a set of input data vectors that will be trained.
- Trained data is a set of data vectors that is trained.
- Untrained data is a set of data vectors that is not yet a trained data.
- Unseen data is a set of data vectors that is used for testing the trained network.

2.2 Radial Basis Function Neural Network (RBF NN)

Radial basis function is a special class of functions. Its characteristic features are the response decreases (or increases) monotonically with the distance from a central point. The center, the distance scale which is the distance from the center that control the width of the function, and the steepness of the radial function are the parameters of the model. A linear model for a function takes the form

$$y = \sum_{j=1}^{n} w_j \varphi_j(\mathbf{x}) \tag{2.3}$$

where j is the index of the radial function, φ is the radial function which all parameters in the function are fixed except the input vector $\mathbf{x}$, w_j is the coefficient of the linear combination of the fixed radial function (φ_j) which is a weight of each function, and y is a response of this linear combination of the set of n fixed functions model. But if the parameters in the radial functions can be changed during the learning process, or the network has more than one hidden layer then the model is nonlinear.

The model is trained by adjusting the center, the size, and the steepness parameters of the radial basis function. A typical radial basis function is the Gaussian distribution function, which, in the case of a scalar input, is defined as follows

$$h(x) = \exp(-\frac{(x-c)^2}{r^2}) \tag{2.3}$$

where x is the input, c is the center point, and r is the variance or width of the function. Figure 2.3 shows a Gaussian RBF shape with center c = 0 and radius r = 1.

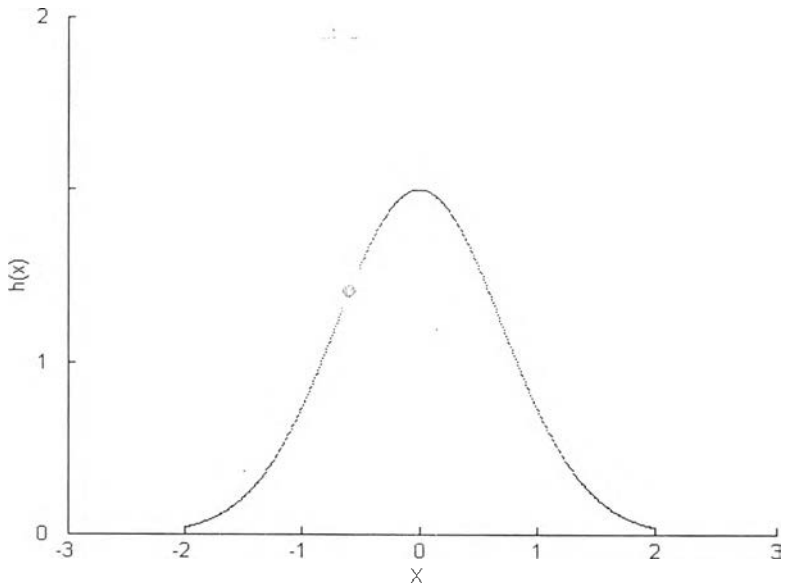

Figure 2.3 Gaussian shape.

In case of input vector, the function is defined as follows

$$h(\mathbf{x}) = \exp(-\frac{\|\mathbf{x}-\mathbf{c}\|^2}{\mathbf{r}^2}) \tag{2.4}$$

where $\mathbf{x}$ is the input vector, $\mathbf{c}$ is the center vector, and $\mathbf{r}$ is the variance or width vector of the function. Figure 2.4 shows the shape of the function in a 2-dimensional space with radius $\mathbf{r}$ = (2,2), center $\mathbf{c}$ = (0,0), and $\mathbf{x}$ is a vector located the coordinates (1,2) in n-dimension space. The two parameters, $\mathbf{r}$ and $\mathbf{c}$, are used to define the radial function's size and location, respectively.

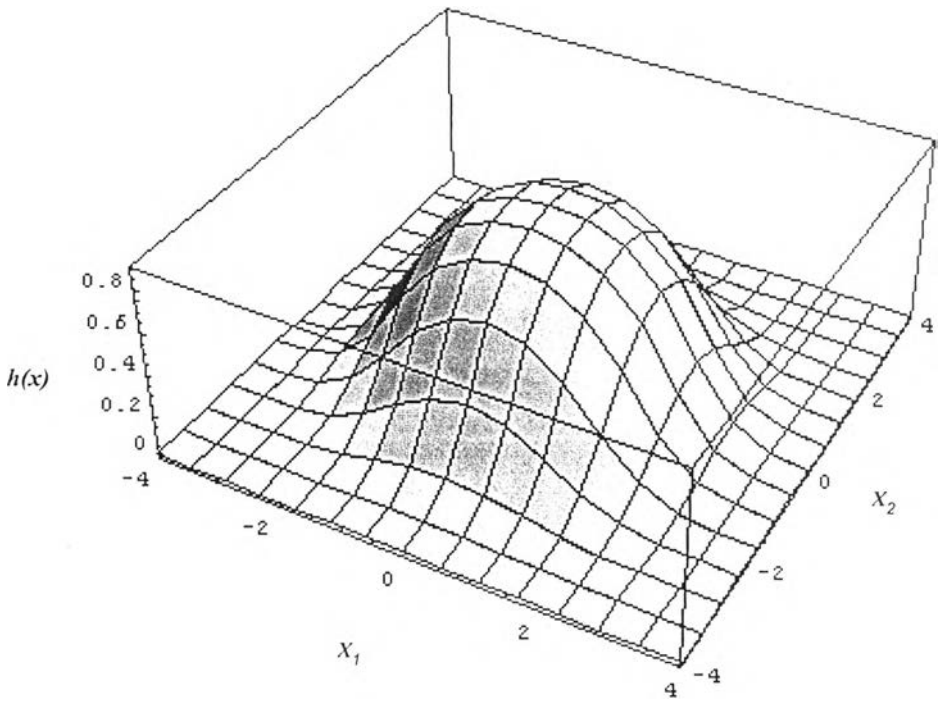

Figure 2.4 Gaussian shape in a 2-dimension space of the input data vectors.

In principle, they could be employed in any sort of model (linear or nonlinear). It is up to the fixing of the parameters in the function. And they could be employed in any sort of network (single-layer or multi-layer) too. However, a radial basis function network has been traditionally associated with radial functions in a single-layer network as shown in Figure 2.5.

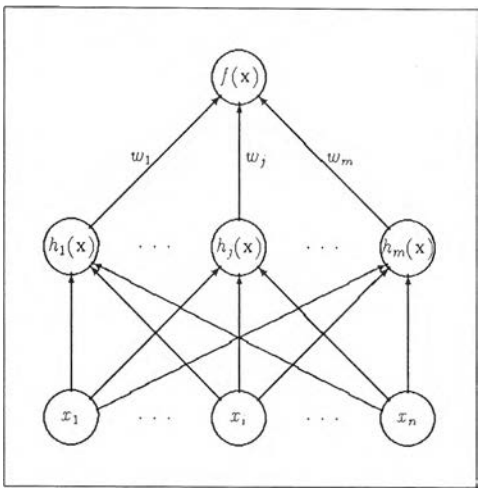

Figure 2.5 The traditional radial basis function neural network. Each of n components of an input vector is fed forward to m basis functions whose outputs are linearly combined with w_j weights to the network output.

Each parameter in Figure 2.5 is defined as follows: $f(x)$ is the function of the output of the network, w_j is the weight for the hidden node j, and $h_j(x)$ is the hidden node function j. Here, we focus on a single-layer network (because the 2-dimentional classification problem can be solved by a single-layer network correctly) with functions that are varied on center vectors and width vectors. The learning algorithm starts with a fixed number of hidden nodes, say m nodes. After that each hidden node is relocated to an appropriate location to cover all members of the same class in training data set.

2.3 Resource Allocating Neural Network (RAN)

This is one kind of Gaussian RBF neural networks that allocates a new hidden node when the network is trained with untrained data vectors. The network can be used without repeated training, which implies that each data vector needs only one time training. RAN is a 2-layer neural network (which this number of layer is counted by

number of hidden and output layer [2]). The hidden nodes are realized by the following Gaussian function:

$$z_j = \sum_k (\mathbf{c}_{jk} - \mathbf{I}_k)^2 \tag{2.5}$$

$$x_j = e^{\left(\frac{-z_j}{\mathbf{w}_j^2}\right)} \tag{2.6}$$

$$\bar{y} = \sum_j \bar{h}_j x_j + \bar{\gamma}. \tag{2.7}$$

where $\mathbf{c}_{jk}$ is a center vector of the Gaussian function (x_j), $\mathbf{w}_j$ is a width vector of the Gaussian function (x_j), $\mathbf{I}_k$ is a input vector k, z_j is an Euclidean distance from $\mathbf{I}_k$ to $\mathbf{c}_{jk}$, x_j is the Gaussian function j, $\bar{h}_j$ is the weight of link j connecting the hidden neuron j and the output neuron (Gaussian), $\bar{\gamma}$ is the bias [1], and $\bar{y}$ is output of the network [2]. The structural of a RAN is shown in Figure 2.5. This network does not need any initial number of hidden nodes as the traditional RBF network. It learns by introducing a node in the hidden layer at first and, then, gradual adding a new RBF node whenever the classification is not successful.

2.4 Bootstrap Method

The Standard Quenouill-Tukey jackknife is the idea of the estimation of the mean and variance of the population [7]. After that it had been developed to be a Bootstrap and the other computer-intensive statistical methods by Efron [7], [8]. The bootstrap calculates the population parameters by the resampling techniques. So the confidence intervals of this method have been widely discussed for using in the general situation [9]. The bootstrap is a resampling procedure that resamples from the original data set. It is used for estimation of the population parameter, μ^B (bootstrap median) or σ^B (bootstrap variance), based on the following procedure.

1. Generate a sample of size n with replacement from the empirical distribution.

2. Compute the statistic parameter $\hat{\mu}$ (in case of finding median of the population), which its value is obtained by using the bootstrap sample (the sample set that resampling from the original data) in place of the original sample.
3. Repeat steps 1 and 2 k times.

We are able to compute the estimator of the median of each sample set, and find mean of them again. The confidence interval estimation depends on k. If k is large enough, the area of the confidence interval is large too.

In the second step we compute the statistic parameter in case of median of the population $\hat{\mu}$ by

$$\hat{\mu}_j = \frac{\sum_{i=1}^{n} \mathbf{x}_i}{n} \tag{2.8}$$

where $\mathbf{X}_i$ is the data vectors, i is the index of the data vectors, n is the number of data vectors, and $\hat{\mu}_j$ is the median of the sample set j. In case of variance estimation ($\hat{\sigma}$), we follow the above three steps. But in the computation of the statistic parameter, we use following equation instead.

$$\hat{\sigma}_j = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{x}_i - \overline{\mathbf{x}})} \tag{2.9}$$

where $\overline{\mathbf{x}}$ is the median of an input data that can be changed to the bootstrap median μ^B when its confidence interval is accepted [9]. When we finish all of above procedure, we average all statistical parameters for estimating the grand statistical parameter of the data set by the following equations

$$\mu^B = \frac{\sum_{j=1}^{k} \hat{\mu}_j}{k} \tag{2.10}$$

$$\sigma^B = \frac{\sum_{j=1}^{k} \hat{\sigma}_j}{k} \tag{2.11}$$

We can follow the bootstrapping by follow the following diagram.

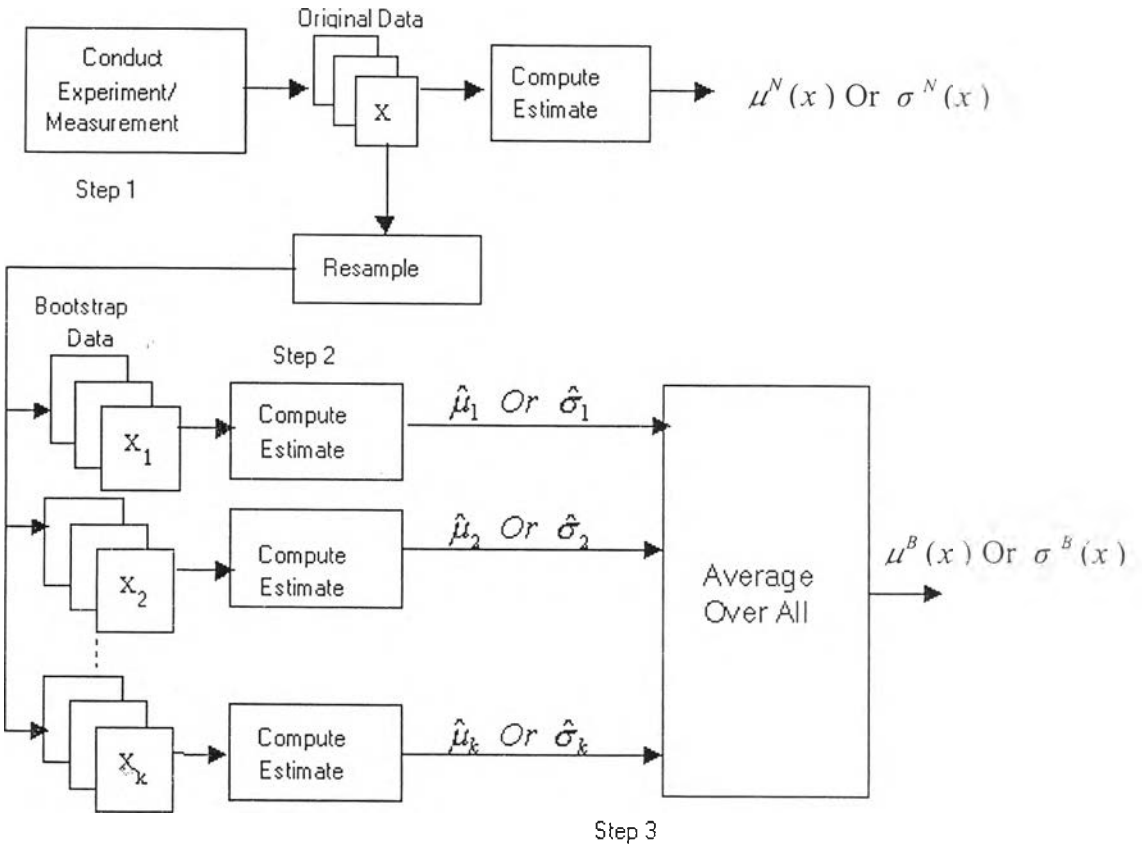

Figure 2.6 The Bootstrap principle.

Figure 2.5 shows the bootstrapping diagram (comparison between normal estimation and bootstrapping estimation) when we want to estimate the population parameter (μ^N Or σ^N) from an original data. According to the normal estimation, if we want to find the population parameters of the data set, we will compute it by considering all data in the set following the three top steps of the diagram. But if we use the bootstrapping estimation, we will follow the other step instead. The diagram starts with a conduction of an experiment or a measurement for creating the original data or the population in step 1. Then, the data flow to either Compute Estimate step (for generating of the normal estimation) or Resample step (for bootstrapping estimation). In the normal estimation, the diagram finishes here, but in the bootstrapping estimation the diagram flow through the Resample step. After Resampling, the *k* sample sets are generated and fed to step 2. In step 2, all data in each sample set are considered for creating each sample set population parameter by equations 2.8 or 2.9. Finally, we create the grand population parameter (μ^B Or σ^B) by averaging all of the sample parameters by equations 2.10 or 2.11. So this diagram presents how bootstrapping does not need the considering of all original data.