Radial-basis function network (RBFN).
Wavelet neural network (Wavelet NN)

Radial-basis function network (RBFN).

Unlike a multilayer network where signal transformation is implemented by combined efforts of many neurons in an arbitrary point of space radial-basis function network (RBFN) represents the input/output of nonlinear systems by adaptation of several single approximate functions to the expected values in a constrained domain of multidimensional space. In using such approach data set mapping is the sum of local transformations.

RBFN is two-layer network with one hidden layer where hidden neurons realize functions radially changing around chosen center and having nonzero values only in the neighborhood of this center. Such functions are called radial-basis and written as:

$$\varphi\left(\left\|\mathbf{X}-\mathbf{C}\right\|\right),$$

where $\mathbf{X}=\left[x_1, x_2, ..., x_M\right]^t$ is the vector of input signals; t is the transposition sign; $\mathbf{C}=\left[c_1, c_2, ..., c_M\right]^t$ is the coordinates vector of function expansion center.

The argument of the activation function of each hidden unit in an RBF network computes the Euclidean norm (distance) between the input vector and the center of that unit.

In most applications the hidden space has dimensionality higher than the dimensionality of the input space.

In RBFN a hidden neuron performance is in mapping of radial space round a single given point or round a group of such points forming a cluster. **Superposition of signals coming from all hidden neurons is performed by the output neuron. Superposition allows to obtain the whole multidimensional space mapping.**

The structure of radial-basis function network is shown in Fig. 6.1.

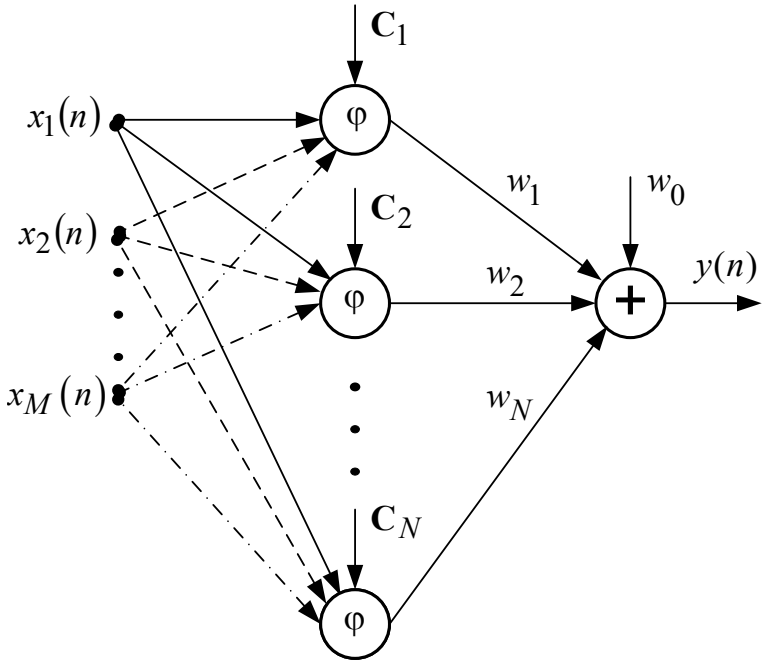

Fig. 6.1. The radial-basis function network structure

Its mathematical model is described by the expression

$$y(n) = w_0 + \sum_{i=1}^{N} w_i \varphi\left(\left\|\mathbf{X}(n) - \mathbf{C}_i\right\|\right), \tag{6.1}$$

where $\mathbf{X}(n) = \left[x_1(n), x_2(n), ..., x_M(n)\right]^t$ is the vector of network input signals; t is the transposition sign; w_i $(i = 1, 2, ..., N)$ is the weight coefficients of output neuron synaptic connections; w_0 is the constant bias; $\mathbf{C}_i$, $\mathbf{C}_i = \left[c_{1i}, c_{2i}, ..., c_{Mi}\right]^t$ is the vector of the i-th center coordinates.

Some examples of radial basis functions in one-dimensional space are depicted in Fig. 6.2.

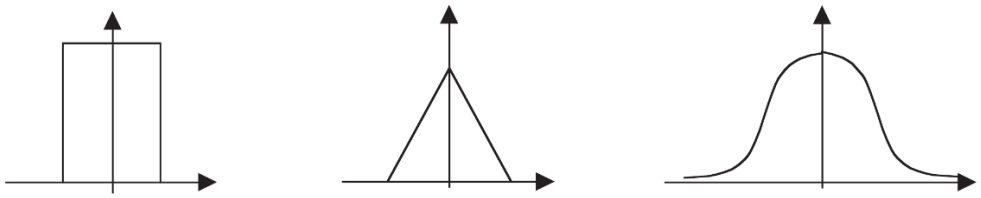

Fig. 6.2. Three examples of one-dimensional radial-basis functions

Radial function φ is often given as Gaussian function in model (6.1)

$$\varphi\left(\left\|\mathbf{X}(n)-\mathbf{C}_i\right\|\right)=\exp\left(-\sum_{m=1}^{M}\frac{\left(x_m(n)-c_{mi}\right)^2}{2\sigma_{mi}^2}\right),$$

where σ_{mi} is the width of Gaussian function relative to element $x_m(n)$; $\mathbf{C}_i=\left[c_{1i},c_{2i},...,c_{Mi}\right]^t$ is the vector of the i-th center coordinates of radial function φ.

When a center is located in $\mathbf{C}_i$ point Gaussian function is defined in a simplified form for $\sigma_{1i}=\sigma_{2i}=...=\sigma_{Mi}=\sigma_i$:

$$\varphi\left(\left\|\mathbf{X}(n)-\mathbf{C}_i\right\|\right)=\exp\left(-\frac{1}{2\sigma_i^2}\sum_{m=1}^{M}\left(x_m(n)-c_{mi}\right)^2\right)=$$

$$=\exp\left(-\frac{1}{2\sigma_i^2}\left\|\mathbf{X}(n)-\mathbf{C}_i\right\|^2\right)=\exp\left(-\frac{1}{h_i}\left\|\mathbf{X}(n)-\mathbf{C}_i\right\|^2\right), \qquad (6.2)$$

where σ_i is the width of Gaussian function, $h_i=2\sigma_i^2$ is a scalar coefficient, that is a parameter influencing the width of function; $\|.\|$ is the Euclidean norm:

$$\left\|\mathbf{X}(n)-\mathbf{C}_i\right\|^2=\left(\mathbf{X}(n)-\mathbf{C}_i\right)^t\cdot\left(\mathbf{X}(n)-\mathbf{C}_i\right)=$$

$$=\left(x_1(n)-c_{1i}\right)^2+\left(x_2(n)-c_{2i}\right)^2+...+\left(x_M(n)-c_{Mi}\right)^2.$$

RBFN drawback is related to interpolation imperfection in the cases when the branches of neighboring Gaussian functions overlap each other. In similar situation there may arise descents between individual Gaussian bell-shaped curves in some parts of neural network model surface. If the distance between the "bells" centers is large and the ranges covered by branches are small, there may arise dips caused by model local nonsensitivity to input changes. The negative effects mentioned are substantially decreased if normalized networks are used.

Normalized radial-basis function network (NRBFN) is described by mathematical model of the form

$$y(n) = w_0 + \frac{1}{\sum_{q=1}^{N} \varphi\left(\left\|\mathbf{X}(n) - \mathbf{C}_q\right\|\right)} \sum_{i=1}^{N} w_i \varphi\left(\left\|\mathbf{X}(n) - \mathbf{C}_i\right\|\right) =$$

$$= w_0 + \sum_{i=1}^{N} w_i \psi\left(\mathbf{X}(n), \mathbf{C}_i\right), \tag{6.3}$$

where $\psi\left(\mathbf{X}(n), \mathbf{C}_i\right)$ is the normalized radial-basis function:

$$\psi\left(\mathbf{X}(n), \mathbf{C}_i\right) = \frac{\varphi\left(\left\|\mathbf{X}(n) - \mathbf{C}_i\right\|\right)}{\sum_{q=1}^{N} \varphi\left(\left\|\mathbf{X}(n) - \mathbf{C}_q\right\|\right)} = \frac{\exp\left(-\frac{1}{2\sigma^2}\left\|\mathbf{X}(n) - \mathbf{C}_i\right\|^2\right)}{\sum_{q=1}^{N} \exp\left(-\frac{1}{2\sigma^2}\left\|\mathbf{X}(n) - \mathbf{C}_q\right\|^2\right)}.$$

The NRBFN structure with the model (6.3) is depicted in Fig. 6.3.

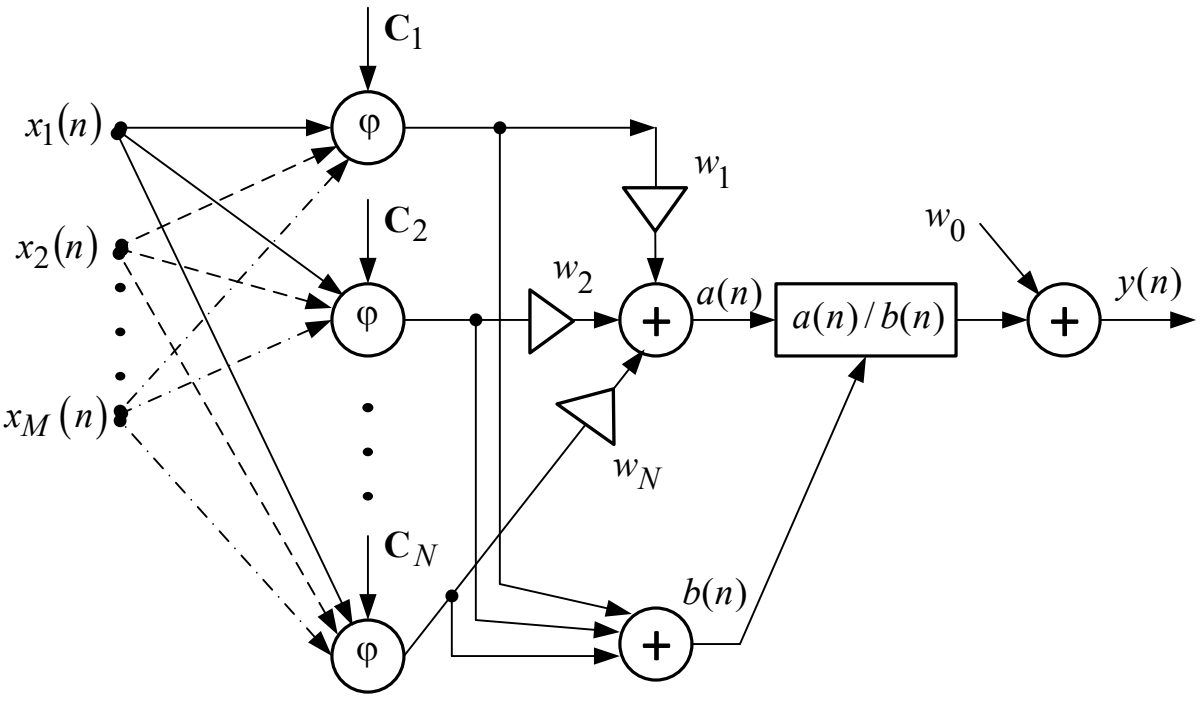

Fig. 6.3. The structure of normalized radial-basis network

In the process of RBFN and NRBFN learning we use back error propagation algorithm, hybrid algorithm the process of self-organization etc.

By force of radial functions RBFN and NRBFN local character the dependence between basis functions parameters and physical location of data for learning multidimensional space is easily defined. As a result, it is rather simple to succeed in obtaining satisfactory initial condition of network process learning.

An important advantage of radial-basis function network is the simplified learning algorithm. If there is only one hidden layer and close connection of neuron activity with the according domain in the space of data for learning, **the point of learning origin appears to be located much closer to the optimal solution than in multi-layer networks. Besides, the stage of basis functions parameters selection is separated from network weights values selection (hybrid algorithm) and this considerably simplifies and accelerates the learning process.**

The complexity of working with radial-basis function network consists in the selection of basis functions quantity, one hidden neuron corresponding to each of them. Small quantity of neurons doesn't allow to decrease generalization error of learning data set, while large quantity of neurons increases computation complexity.

Radial-basis functions can be used to approximate an assigned function. For example, as illustrated in Fig. 6.4, a rectangular-shaped radial-basis function can be used to construct a staircase approximation of a function, and a triangular-shaped radial-basis function can be used to construct a trapezoidal approximation of a function.

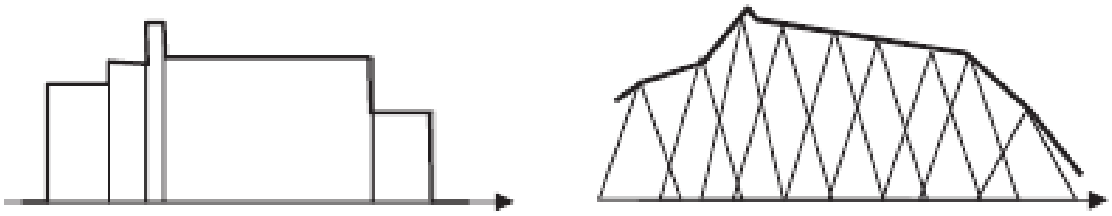

Fig. 6.4. Two examples illustrating radial-basis function approximation

In each of the examples in Fig. 6.4, the approximated function can be represented as a weighted linear combination of a family of radial-basis functions with different scaling and translations:

$$\tilde{F}(\mathbf{X}) = w_0 + \sum_{i=1}^{N} w_i \varphi\left(\left\|\mathbf{X} - \mathbf{C}_i\right\|\right) \tag{6.4}$$

This function is realized with a radial-basis network, as shown in Fig. 6.1.

There are three types of radial-basis networks based on how the radial-basis functions are placed and shaped.

To introduce these radial-basis networks, let us present the function approximation problem formulation. Given set of points $\{\mathbf{X}(k); 1 \le k \le K\}$ and the values of an unknown function $F(\mathbf{X})$ evaluated on these K points $\{d(k) = F(\mathbf{X}(k)); 1 \le k \le K\}$, find an approximation of $F(\mathbf{X})$ in the form of equation (6.4) such that the sum of square approximation error at these sets of training samples,

$$\sum_{k=1}^{K} \left[d(k) - \tilde{F}\left(\mathbf{X}(k)\right) \right]^2$$

is minimized.

Type I Radial-Basis Network (Fixed Centers Selected at Random)

The simplest approach is to assume fixed radial-basis functions defining the activation functions of the hidden units. The locations of the centers may be chosen randomly from the training data set. In other words, it sets $\mathbf{C}_i = \mathbf{X}(i)$, where $1 \le i \le N$.

Furthermore, a fixed constant scaling parameter σ is chosen for every radial-basis function. For the radial-basis functions themselves, we may employ an **isotropic Gaussian function** whose standard deviation is fixed according to the spread of the centers. Specifically, a (normalized) radial-basis function centered at $\mathbf{C}_i$ is defined as

$$\varphi\left(\left\|\mathbf{X}(n) - \mathbf{C}_i\right\|\right) = \exp\left(-\frac{N}{d_{\max}^2} \left\|\mathbf{X}(n) - \mathbf{C}_i\right\|^2 \right), \; i = 1, 2, ..., N,$$

where N is the number of centers and $d_{\max}$ is the maximum distance between the chosen centers. In effect, the standard deviation (i.e., width) of all the Gaussian radial-

basis functions is fixed at

$$\sigma = \frac{d_{\max}}{\sqrt{2N}}.$$

Now rewrite (6.4) in a vector inner product formulation:

$$\left[1 \;\; \varphi\left(\left\|\mathbf{X}(k)-\mathbf{C}_1\right\|\right) \varphi\left(\left\|\mathbf{X}(k)-\mathbf{C}_2\right\|\right) \ldots \varphi\left(\left\|\mathbf{X}(k)-\mathbf{C}_N\right\|\right)\right]\begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \ldots \\ w_N \end{bmatrix} = d(k). \qquad (6.5)$$

Substituting $k = 1, 2, ..., K$, (6.5) becomes a matrix equation $\mathbf{\Phi W} = \mathbf{D}$:

$$\begin{bmatrix} 1 & \varphi\left(\left\|\mathbf{X}(1)-\mathbf{C}_1\right\|\right) & \varphi\left(\left\|\mathbf{X}(1)-\mathbf{C}_2\right\|\right) & \ldots & \varphi\left(\left\|\mathbf{X}(1)-\mathbf{C}_N\right\|\right) \\ 1 & \varphi\left(\left\|\mathbf{X}(2)-\mathbf{C}_1\right\|\right) & \varphi\left(\left\|\mathbf{X}(2)-\mathbf{C}_2\right\|\right) & \ldots & \varphi\left(\left\|\mathbf{X}(2)-\mathbf{C}_N\right\|\right) \\ & \ldots & \ldots & & \ldots \\ 1 & \varphi\left(\left\|\mathbf{X}(K)-\mathbf{C}_1\right\|\right) & \varphi\left(\left\|\mathbf{X}(K)-\mathbf{C}_2\right\|\right) & \ldots & \varphi\left(\left\|\mathbf{X}(K)-\mathbf{C}_N\right\|\right) \end{bmatrix}\begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \ldots \\ w_N \end{bmatrix} = \begin{bmatrix} d(1) \\ d(2) \\ \ldots \\ d(K) \end{bmatrix}.$$

$\mathbf{\Phi}$ is a $K \times (N+1)$ square matrix (note that $K \geq N$), and is generally positive for commonly used radial-basis functions.

However, in practical applications, the $\mathbf{\Phi}$ matrix may be nearly singular, leading to a numerically unstable solution of $\mathbf{W}$. This can happen when two or more samples $\mathbf{X}(k)$s are too close to each other. Several different approaches can be applied to alleviate this problem.

Method 1: Regularization. For a small positive number λ, a small diagonal matrix is added to the radial-basis coefficient matrix $\mathbf{\Phi}$ such that

$$\mathbf{W} = \left(\mathbf{\Phi} + \lambda \mathbf{I}\right)^{-1}\mathbf{D}.$$

Method 2: Least Square Using Pseudo-Inverse. The goal is to find a least square solution $\mathbf{W}_{LS}$ such that $\left\|\mathbf{\Phi W} - \mathbf{D}\right\|^2$ is minimized. Hence,

$$\mathbf{W} = \mathbf{\Phi}^{+}\mathbf{D},$$

where $\mathbf{\Phi}^{+}$ is the pseudo-inverse matrix of $\mathbf{\Phi}$ and can be found using singular value decomposition (SVD).

Type II Radial-Basis Network (Self-Organized Selection of Centers)

The main problem with the method of fixed centers just described is the fact that it may require a large training set for a satisfactory level of performance. One way of overcoming this limitation is to use a hybrid learning process, consisting of two different stages:

- **Self-organized learning stage,** the purpose of which is to estimate appropriate locations for the centers of the radial basis functions in the hidden layer.
- **Supervised learning stage,** which completes the design of the network by estimating the linear weights of the output layer.

For the self-organized learning process we need a clustering algorithm that partitions the given set of data points into subgroups, each of which should be as homogeneous as possible. **One such algorithm is the *k*-means clustering algorithm, which places the centers of the radial-basis functions in only those regions of the input space where significant data are present.** Let N denote the number of radial-basis functions; the determination of a suitable value for N may require experimentation. Let $\{\mathbf{C}_k(l)\}_{k=1}^{N}$ denote the centers of the radial-basis functions at iteration *l* of the algorithm. Then, the *k*-means clustering algorithm proceeds as follows:

1. Initialization. Choose random values for the initial centers $\mathbf{C}_k(0)$; the only restriction is that these initial values be different. It may also be desirable to keep the Euclidean norm of the centers small.

2. Sampling. Draw a sample vector $\mathbf{X}$ from the input space with a certain probability. The vector $\mathbf{X}$ is input into the algorithm at iteration *l*.

3. Similarity matching. Let $k(\mathbf{X})$ denote the index of the best-matching (winning) center for input vector $\mathbf{X}$. Find $k(\mathbf{X})$ at iteration *l* by using the minimum-distance Euclidean criterion:

$$k(\mathbf{X}) = \arg\min_k \|\mathbf{X}(l) - \mathbf{C}_k(l)\|, \quad k = 1, 2, ..., N,$$

where $\mathbf{C}_k(l)$ is the center of the *k*-th radial-basis function at iteration *l*.

4. Updating. Adjust the centers of the radial-basis functions, using the update rule:

$$\mathbf{C}_k(l+1)=\begin{cases}\mathbf{C}_k(l)+\eta\left[\mathbf{X}(l)-\mathbf{C}_k(l)\right], k=k(\mathbf{X}),\\ \mathbf{C}_k(l)\ \text{ otherwise,}\end{cases}$$

where η is a learning-rate parameter that lies in the range $0<\eta<1$.

5. Continuation. Increment *l* by 1, go back to step 2, and continue the procedure until no noticeable changes are observed in the centers $\mathbf{C}_k$.

The *k*-means clustering algorithm just described is, in fact, a special case of a competitive (winner-takes-all) learning process known as the self-organizing map.

Having identified the individual centers of the Gaussian radial-basis functions and their common width using the *k*-means clustering algorithm or its enhanced version, the next and final stage of the hybrid learning process is to estimate the weights of the output layer. A simple method for this estimation is the least-mean-square (LMS) algorithm.

A limitation of the *k*-means clustering algorithm is that it can only achieve a local optimum solution that depends on the initial choice of cluster centers. Consequently, an unnecessarily large network may be created.

Type III Radial-Basis Network (Supervised Selection of Centers)

In the third approach, **the centers, widths and weights of the network undergo a supervised learning process**; in other words, the RBF network takes on its most generalized form. A natural candidate for such a process is error-correction learning, which is most conveniently implemented **using a gradient-descent procedure** that represents a generalization of the LMS algorithm.

The first step in the development of such a learning procedure is to define the cost function

$$E=\frac{1}{2}\sum_{k=1}^{K}e_k^2,$$

where *K* is the size of the training set and e_k is the error signal defined by

$$e_k=d(k)-\tilde{F}(\mathbf{X}(k))=d(k)-\sum_{i=0}^{N}w_i\varphi\left(\left\|\mathbf{X}(k)-\mathbf{C}_i\right\|\right),$$

where $\varphi\left(\left\|\mathbf{X}(0)-\mathbf{C}_i\right\|\right)=1$ is the output from the bias neuron,

$$\varphi\left(\left\|\mathbf{X}(k)-\mathbf{C}_i\right\|\right)=\exp\left(-\frac{1}{2\sigma_i^2}\sum_{m=1}^{M}\left(x_m(k)-c_{mi}\right)^2\right)$$

is Gaussian function.

The requirement is to find the free parameters $\mathbf{W},\mathbf{C},\boldsymbol{\sigma}$ so as to minimize *E*.

Unlike the back-propagation algorithm, **the gradient-descent procedure described below for RBF network does not involve error back-propagation.**

The update equations for $\mathbf{W},\mathbf{C},\boldsymbol{\sigma}$ are (in general) assigned different learning-rate parameters $\eta_w,\eta_c,\eta_\sigma$ respectively.

Adaptation formulas for the linear weights, the positions and spreads of centers for RBF Network are the following:

1. Linear weights (output layer)

$$\frac{\partial E(l)}{\partial w_i(l)}=\sum_{k=0}^{K}e_k(l)\varphi\left(\left\|\mathbf{X}(k)-\mathbf{C}_i(l)\right\|\right),$$

$$w_i(l+1)=w_i(l)-\eta_w\frac{\partial E(l)}{\partial w_i(l)},\ i=0,1,...,N.$$

2. Positions of centers (hidden layer)

$$\frac{\partial E(l)}{\partial \mathbf{C}_i(l)}=\sum_{k=0}^{K}e_k(l)w_i(l)\varphi\left(\left\|\mathbf{X}(k)-\mathbf{C}_i(l)\right\|\right)\frac{\left(\mathbf{X}(k)-\mathbf{C}_i(l)\right)}{\sigma_i^2},$$

$$\mathbf{C}_i(l+1)=\mathbf{C}_i(l)-\eta_c\frac{\partial E(l)}{\partial \mathbf{C}_i(l)},\ i=1,2,...,N$$

3. Spreads of centers (hidden layer)

$$\frac{\partial E(l)}{\partial \sigma_i(l)}=\sum_{k=0}^{K}e_k(l)w_i(l)\varphi\left(\left\|\mathbf{X}(k)-\mathbf{C}_i(l)\right\|\right)\frac{\left(\mathbf{X}(k)-\mathbf{C}_i(l)\right)\left(\mathbf{X}(k)-\mathbf{C}_i(l)\right)^t}{\sigma_i^3},$$

$$\sigma_i(l+1)=\sigma_i(l)-\eta_\sigma\frac{\partial E(l)}{\partial \sigma_i(l)},\ i=1,2,...,N.$$

Wavelet neural network

Non-linear dynamic system simulation is effectively implemented on the base of **wavelet neural networks (wavelet NN) containing mother wavelet described by expression**

$$\Psi_{a,b}(x) = \frac{1}{\sqrt{a}}\Psi\left(\frac{x-b}{a}\right),$$

where a, b are scaling and shift parameters respectively; $\Psi(x)$ is the wavelet function meeting the condition

$$C_\psi = \int_{-\infty}^{\infty} \frac{\left|\widetilde{\Psi}(\omega)\right|^2}{|\omega|} d\omega < \infty,$$

$\widetilde{\Psi}(\omega)$ is the Fourier-transform of $\Psi(x)$.

Wavelet neural network is the universal approximator for non-linear dynamic systems operators. Wavelet NN is represented as a structure depicted in Fig. 6.5 in the appropriate mathematical form

$$y(n) = \sum_{i=1}^{N} v_i \Psi_i\left(\frac{c_i - b_i}{a_i}\right) = \sum_{i=1}^{N} v_i \Psi\left(\frac{c_i - b_i}{a_i}\right) =$$

$$= \sum_{i=1}^{N} v_i \Psi\left(\frac{\left[\sum_{j=1}^{m} w_{ij} x_j(n)\right] - b_i}{a_i}\right),$$

where $y(n)$ is the model output signal; N is the number of neurons; $c_i = \sum_{j=1}^{m} w_{ij} x_j(n)$; $x_j(n)$ is the j-th model input signal contained in excitation vector $\mathbf{X}(n) = \left[x_1(n), x_2(n), ..., x_j(n), ..., x_m(n)\right]^t$; Ψ is the wavelet function ($\Psi = \Psi_1 = ... = \Psi_N$).

Variables w_{ij}, a_i, b_i, v_i ($i = 1, 2, ..., N$, $j = 1, 2, ..., m$) are model parameters.

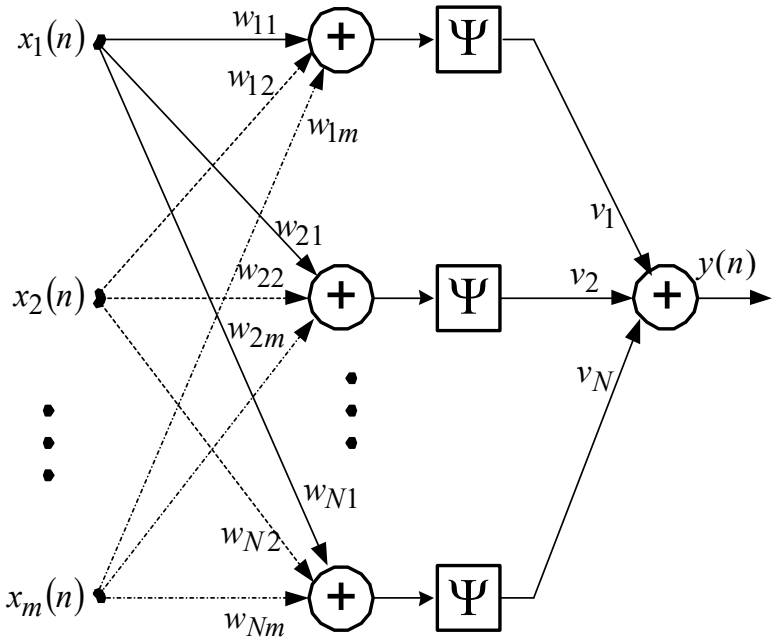

Fig. 6.5. The wavelet neural network structure

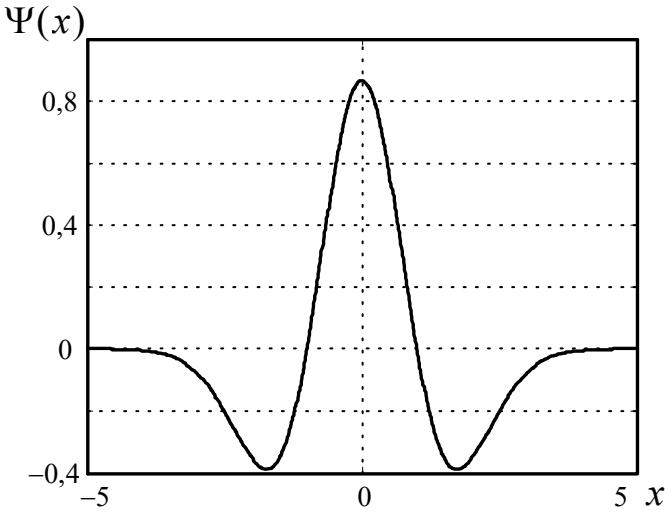

Fig. 6.6. The plot of mother wavelet “Mexican hat”

The set of wavelet functions Ψ is highly broad (Haar wavelet, Morlet wavelet, Meyer wavelet, Mallat wavelet, Daubechies wavelet, Mexican hat (Ricker) wavelet and so on). As an example we give mother wavelet “Mexican hat”:

$$\Psi(x) = \left(\frac{2}{\sqrt{3}}\pi^{-0,25}\right)(1-x^2)\exp\left(-\frac{x^2}{2}\right),$$

the plot of which is shown in Fig. 6.6.