A survey on hyper basis function neural networks
Yuguo Zhou, Tong Mu, Zhong-Hua Pang and Changbing Zheng

School of Information and Control Engineering, Qingdao University of Technology, Qingdao, People’s Republic of China; Key Laboratory of Fieldbus Technology and Automation of Beijing, North China University of Technology, Beijing, People’s Republic of China

ABSTRACT
Hyper basis function neural networks (HBFNNs) have gained considerable attention in recent years, which have shown good performance in a variety of application domains. In this paper, we first briefly introduce the development of neural networks. Then the structure of HBFNNs is presented in detail. HBFNNs are an extension of radial basis function neural networks (RBFNNs), which use the weighted norm instead of the Euclidean norm to represent the distance from input data to hidden layer neuron centres. With this change, the generalization ability of neural networks becomes stronger than that of RBFNNs. Subsequently, we summarize several commonly used training methods for HBFNNs, including static training methods and dynamic training methods. Finally, we give several typical application fields of HBFNNs.

1. Introduction
In recent years, with the growth of artificial intelligence technologies, the research on artificial neural networks has been paid much more attention. As one of the most popular multi-layer feed-forward neural networks, radial basis function neural networks (RBFNNs) have many advantages such as simple networks structure, fast training, and global approximation capability with local responses, which have demonstrated excellent performance in various application fields (Dash, Behera, Dehuri, & Cho, 2016). Poggio and Girosi (1990) proposed a hyper basis function (HBF), which was a generalization of the radial basis function (RBF). Then they presented a hyper basis function neural network (HBFNN) as a regularization network, which replaced the Euclidean norm used in RBFNNs with a weighted norm for the activation function in the hidden layer, such that the distance between input data and hidden layer neuron centres was described in a high degree of freedom form. Through this approach, HBFNNs can decrease the correlation interference between input variables (Wen, Zhang, & Zhu, 2015). Moreover, the number of HBFNN neurons needed is smaller than that of a RBFNN in order to learn a complex function, which makes HBFNNs more competitive to achieve fast learning with limited computation resources (Mahdi & Rouchka, 2011; Vukovi & Miljkovi, 2013; Wu, Kong, & Yang, 2018).

However, there were several problems in original HBFNNs to limit their applications, for example (Mahdi & Rouchka, 2011; Vukovi & Miljkovi, 2013):

(1) The high freedom degree of HBFNNs is easy to lead to overfitting and poor generalization.
(2) HBFNNs tend to converge towards locally optimal rather than globally optimal solutions.
(3) Training an HBFNN is a challenging problem which demands a scalable optimization method to estimate the large number of parameters.

In view of the above shortcomings of HBFNNs, different training methods have been proposed for HBFNNs. Mahdi and Rouchka (2011) proposed a new regularization algorithm to train HBFNNs. When dealing with classification problems, the classification accuracy of HBFNNs was comparable to the support vector machine (SVM) approach. In addition, the intelligent selection of neuron centres and the dynamic increase or decrease of the hidden neuron number were also adopted in the training of HBFNNs. Especially, HBFNNs have an advantage in solving the problems with small data, complex nonlinearity and high precision requirement. Therefore, HBFNNs have become one of the hotspots of feedforward neural networks research. Nowadays, HBFNNs have been widely used in pattern recognition (Lu, 2008), intelligent
robot (Rao, Ramji, Rao, Vasu, & Puneeth, 2017; Yang & Tan, 2018), image processing (Ha et al., 2018), biology and medicine (Nieminen, Hakama, Viikki, Tarkkanen, & Anttila, 2003; Vicente et al., 2012), economy (Kimoto, Asakawa, Yoda, & Takeoka, 1990) and other fields.

In this paper, a survey is given for HBFNNs. We briefly summarize the development of neural networks in Section 2. The structure and main training methods of HBFNNs are discussed in Sections 3 and 4, respectively. Then some practical applications of HBFNNs are introduced in Section 5. Finally, we conclude this paper in Section 6.

2. Development process of neural networks

Since the birth of neural networks in the 1940s, their development process has experienced three peaks and two downturns, as shown in Figure 1.

1. **The birth of neural networks**: In 1943, McCulloch and Pitts (1943) mathematically modelled biological neurons, and proposed a multi-input and single-output MP neural network model. The MP neural network model has been significantly improved and extended nowadays. The advent of the MP model marked the birth of artificial neural networks.

2. **The first peak period (1952–1969)**: In 1952, Hebb proposed a learning algorithm to adjust weights of neural networks. Moreover, Hebb pointed out that if two neurons were active at the same time, the connection weight between the two neurons would be strengthened. Otherwise, the connection weight would be weakened. Rosenblatt (1958) proposed a forward neural network with a single-layer computing unit – single-layer perceptron. The single-layer perceptron is one of the simplest neural networks with an input layer and an output layer. The two layers are directly connected. Each input signal is weighted, and the weight is adjusted by a learning algorithm. After the single-layer perceptron was proposed, the research on neural networks achieved the first peak, and in the following 10 years, the neural networks were studied in the form of a perceptron.

3. **The first downturn period (1969–1986)**: In 1969, Minsky and Papert proved that a single-layer perceptron had a limitation which cannot solve an XOR problem. Minsky and Papert made the conclusion that ‘perception-based research is doomed to failure’. The research on neural networks reached a downturn. Due to the limited research environment at that time, the research on neural networks almost came to a standstill in the following about 20 years.

4. **The second peak period (1986–1995)**: The above problem of neural networks was solved in the 1980s. The research on neural networks began to be paid much attention on once again. In 1974, Werbos proposed a relatively effective error back-propagation algorithm for solving the XOR problem.

![The development of neural networks](image)

**Figure 1.** The development of neural networks.
(Werbos, 1974), namely the BP algorithm. The error between the predicted and actual outputs obtained from the model and the practical system, respectively, was used to update the weights between layers. In 1986, Hinton and Rumelhart et al. improved the BP algorithm (Rumelhart, Hinton, & Williams, 1988), which greatly reduced the time required for model training. After the BP algorithm was proposed, neural networks were once again taken seriously into consideration. In 1988, Broomhead and Lowe used a RBF to design a multilayer feedforward network, i.e. RBFNN (Broomhead & Lowe, 1988). Compared with BP neural networks, RBFNNs have advantages of fast training speed and high precision. As a result, an efficient neural network was born. In 1990, Poggio and Girosi (1990) extended the RBF by using a regularization theory, and proposed a neural network with a similar network structure and faster training speed, i.e. HBFNN. The research on feedforward neural networks reached another peak.

(5) The second downturn period (1995–2006): Cortes and Vapnik (1995) proposed a machine learning method – support vector machine (SVM), which was an efficient learning algorithm without a local optimal problem. Moreover, Svozil, Kvasnicka, and Pospichal (1997) pointed out that a multilayer feedforward neural network had the disadvantages of local optimization and over-fitting when the number of neuron layers increased. Many researchers on neural networks gradually turned to the study of SVM methods. Before 2006, the SVM approach was considered to be the most successful algorithm in the field of machine learning. The research on multilayer feedforward neural networks gradually became relatively less and once again fell into another downturn.

(6) The third peak period (2006–present): In 2006, Hinton et al. proposed a multilayer neural network – deep confidence network (DBN) and then applied the BP algorithm to the DBN for the first time. After this, the research on multi-layer neural networks boomed once again. In the same year, Hinton, Osindero, and Teh (2006) proposed a deep learning algorithm and a deep neural network (DNN), and then the research on neural networks began to renew a new life in ‘depth’. A convolutional neural network (CNN) was proposed by Lecun in 1989, named LeNet, where Sigmoid was used as a nonlinear activation function. LeNet was the first convolutional neural network, which was successfully applied to digital recognition. After deep learning was proposed, CNNs were combined with deep learning algorithms to produce various deep convolutional neural networks (Zhou & Jiang, 2015). The first deep convolutional neural network was Krizhevsky’s AlexNet convolutional neural network, which won the ImageNet Visual Identity Competition (Krizhevsky, Sutskever, & Hinton, 2012). The AlexNet convolutional neural network used ReLU as a nonlinear activation function and the resulting performance exceeded that of Sigmoid in deeper networks, because ReLU successfully solved the gradient dispersion problem of Sigmoid when the network was deep. As a kind of deep neural networks, deep convolutional neural networks have excellent performance in the fields of face recognition, speech recognition, image processing, etc., which makes deep neural networks become a hotspot of neural network research. In 2014, Szegedy et al. proposed the GoogleNet network structure (Szegedy et al., 2015), which increased the depth and width of the network while reducing parameters. In 2015, Sun led the Microsoft Vision team to propose a deep residual neural networks (ResNet). Poggio, Mhaskar, Rosasco, Miranda, and Liao (2017) explained that deep neural networks could solve the problem of dimensional curse. In 2017, Chen and Liu proposed a new learning framework that was comparable to deep learning – width learning, which solved the problem that deep learning was easy to fall into local optimization, and thus injected a new impetus into the development of neural networks. It is conjectured that in the future, neural networks will have a broader space in the theory development and practical application.

3. Structure of HBFNNs
HBFNNs are a generalization of RBFNNs. The structure of the two classes of neural networks is consistent with the connection approach of neurons among layers. The difference is the expression of the distance from input data to neuron centres in the activation function of the hidden layer.

3.1. RBFNNs
The structure of RBFNNs is shown in Figure 2. A RBFNN usually has three layers which are the input layer, hidden layer and output layer. Input data enter the neural network through the input layer. The neuron activation function of the hidden layer is a RBF. The most commonly used RBF is the Gaussian basis function as follows:

\[ h_j(x) = \exp \left( -\frac{\|x - c_j\|^2}{2\sigma_j^2} \right) \] (1)
The Gaussian basis function peaks to one when the input data vector $x$ is close to the $j$th centre $c_j$ (where the closeness is expressed by similarity that can be defined by the Euclidean distance $\|x - c_j\|$), and decreases to zero in the opposite case (Vuković & Miljković, 2013). The Gaussian basis function can be regarded as a similarity measure between the input data vector $x$ and the centre vector $c_j$. As the Euclidean distance $\|x - c_j\|$ increases, the Gaussian basis function output decreases. In the opposite case, when the Euclidean distance $\|x - c_j\|$ decreases, the output value of the Gaussian basis function is close to one. In other words, greater similarity between input data vector $x$ and centre vector $c_j$ implies bigger output of the Gaussian basis function. This feature is one of the reasons why RBFNNs with Gaussian basis functions are a popular choice for modelling problems in regression, classification, system identification and signal processing. Here, we briefly introduce the similarity (Chen, Wang, & Wang, 2003). The common method of calculating similarity between samples is to calculate the distance between samples. The closer the distance between samples, the higher their similarity. Common similarity expressions are Euclidean distance, Mahalanobis distance, Hamming distance, cosine similarity and so on. Since this paper only uses the Euclidean distance and Mahalanobis distance, we will only introduce the expressions of the two.

The Euclidean distance between two $n$-dimensional vectors $a = [a_1, a_2, ..., a_n]^T$ and $b = [b_1, b_2, ..., b_n]^T$ is expressed as follows:

$$d(a, b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \cdots + (a_n - b_n)^2}$$

(3)

Equation (3) is written as a vector operation as follows:

$$d(a, b) = \sqrt{(a - b)^T S^{-1}(a - b)}$$

(4)

The Mahalanobis distance between two $n$-dimensional vectors $a$ and $b$ is written as a vector operation as follows:

$$d(a, b) = \sqrt{(a - b)^T S^{-1}(a - b)}$$

(5)

where $S$ is the covariance matrix among samples. The similarity of the two vectors is defined as follows:

$$s(a, b) = 1/(1 + d(a, b))$$

(6)

If $S$ is a unit matrix, the Mahalanobis distance also becomes the Euclidean distance. The advantage of the Mahalanobis distance compared to the Euclidean distance is that the correlation interference between the variables is included.

The output of RBFNNs is as follows:

$$y = \sum_{j=1}^{m} w_j h_j(x)$$

(7)

where $w_j$ is the weight of the $j$th hidden layer node to the output node and $y$ is the neural network output.

### 3.2. An extension of RBFNNs – HBFNNs

Poggio and Girosi (1990) used the regularization theory to extend RBFNNs and proposed HBFNNs by using a Mahalanobis-like distance instead of the Euclidean distance. That is, the distance from the input data to the centre of the hidden layer in HBFNNs is represented by a weighted norm, which is different from the European norm in RBFNNs. The activation function of the hidden layer in HBFNNs is as follows:

$$h_j(x) = \exp \left(-0.5 \|x - c_j\|^2_{\Sigma_j} \right)$$

$$= \exp \left(-0.5(x - c_j)^T \Sigma_j(x - c_j) \right)$$

(8)

where $\Sigma_j$ is a positive-definite weighted matrix. According to different requirements, the matrix $\Sigma_j$ can be
divided into different forms (Mahdi & Rouchka, 2011) as follows:

1. All neurons have a spherical shape of the same size: \( \Sigma_j = I/\sigma^2 \), \( I \) is the identity matrix.
2. All neurons have a spherical shape but differ in their size: \( \Sigma_j = 1/\sigma_j^2 \). This is often used for RBF.
3. Every neuron has an elliptical shape with a varying size and orientation: \( \Sigma_j \) is a positive-definite squared matrix that is not diagonal.
4. Every neuron has an elliptical shape with a varying size, but with restricted orientation: \( \Sigma_j = \text{diag}(1/\sigma_{j1}^2, 1/\sigma_{j2}^2, \ldots, 1/\sigma_{jn}^2) \)

   The structure of cases 1 and 2 are oversimplified while case 3 is the most common, because case 3 takes into account the local correlation among dimensions. But it cannot be estimated in the calculation. Due to the excessive degree of freedom of the model, it may lead to serious overfitting. Case 4 is a good balance between the extreme generality of case 3 and the oversimplification of cases 1 and 2. The diagonal elements in case 4 can be interpreted as local scaling factors, ensuring that case 4 is invariant with respect to the local scaling of the dimensions (Schwenker, Kestler, & Palm, 2001). Therefore, case 4 is commonly used in HBFNNs, where the matrix \( \Sigma_j \) is a diagonal matrix.

   We note that the activation function of HBFNNs is expressed in a similar way to the Mahalanobis distance, but not the true Mahalanobis distance. Therefore, we use a concept of Mahalanobis-like distance in the above introduction to describe the basis function of HBFNNs. HBFNNs are not interfered by the correlation between input variables, which makes neural networks have a high degree of freedom and a faster training speed (Vuković et al., 2013; Wu et al., 2018). In addition, we note that when the matrix \( \Sigma_j \) is a covariance matrix, the neural network in this case is called a hyperellipsoidal neural network (Mao & Jain, 1996; Minsky & Papert, 1969; Su & Liu, 2001). The covariance matrix is in the form of a diagonal matrix, and the diagonal elements are the covariance of the input data and the hidden layer centres.

   Compared with RBFNNs, the main advantage of HBFNNs is that the number of neurons required for learning complex functions is small and the training speed is fast. However, HBFNNs have some shortcomings at the same time. For example, HBFNNs have a high degree of freedom and more adjustable parameters. Those shortcomings can make HBFNNs easy to cause over-fitting. However, in the face of small data sets, the performance of HBFNNs is generally better than that of RBFNNs.

4. Training algorithms of HBFNNs

   The training of HBFNNs can be divided into static training and dynamic training (Karayiannis & Mi, 1997; Schwenker et al., 2001). Static training is divided into two steps. The first is to determine the network structure, and the second is to estimate the parameters. Dynamic training refers to another strategy to dynamically add or remove hidden layer neurons in the training process so as to obtain an appropriate neural network model.

4.1. Static training

   For the static training of an HBFNN, the network structure is determined in advance, and then the network parameter training is performed. Network parameters include the neuron centres and the weighted matrices in the hidden layer and the connection weights from the hidden layer to the output layer. Typical network parameter training methods are introduced as follows.

4.1.1. Self-organizing selection centre method

   The self-organizing selection centre method is mainly divided into two stages. The first stage is unsupervised learning. The hidden layer basis function centre vector is determined according to the statistical characteristics of the selected input samples. The second stage is supervised learning. After determining the centre vector, the weight between the hidden layer and the output layer is determined by the least mean square algorithm (LMS) according to the sample training set.

   From the input samples, the clustering method can be used to select the centres of the HBFNNs from the data. The clustering algorithm is an unsupervised learning algorithm. Unsupervised learning means that the marker information of the training sample is unknown, and the intrinsic properties and laws of the data are revealed by learning unknown training samples. Clustering can more accurately reflect the distribution centres of data points.

   The idea of the K-Means algorithm is to divide the sample set into \( K \) clusters for a given sample set according to the distance between the samples. Let the points in the cluster be connected as closely as possible, and make the distance between the clusters as large as possible. This is a typical distance-based clustering algorithm (Bradley & Fayyad, 1998; Kanungo et al., 2002; Likas, Vlassis, & Verbeek, 2003) using distance as the evaluation index of similarity. The closer the distance between two objects, the greater the similarity. The specific process of the K-means algorithm is divided into three steps. First, select the \( K \) points as the initial cluster centre. Second, calculate the distance of each input data to each cluster centre, then assign the data to the category closest to its cluster centre.
and adjust it to calculate new cluster centre. Third, if the cluster centre has not changed twice, it indicates that the data object adjustment ends and the clustering standard function has converged.

The K-means method has two drawbacks (Ding & He, 2004): first, there is an iterative operation during the running of the algorithm, resulting in too a long convergence time. Second, it is impossible to automatically determine the number of HBF centres, so that the hidden layer establishment time is too long. The K-means algorithm is used to determine the hidden layer centre, which improves the accuracy of initializing the centre of the neuron. But it will fall into the local minimum, resulting in a suboptimal solution. According to this shortcoming, Bezdekzai proposed an improved unsupervised learning method – the fuzzy C-means clustering method (Zhang & Jiang, 2009). Although this algorithm has higher performance, the initialization speed is slower and very sensitive to the initialization of the centre. So, it is not suitable for high-dimensional feature vectors. Eberhart and Kennedy (1995) proposed the particle swarm optimization (PSO) algorithm. PSO is an adaptive evolutionary computation technique based on population search. The algorithm is initially inspired by the regularity of the activity of flying birds and fish swarms, and uses group intelligence to build a simplified model.

In the PSO algorithm, the potential solution of each optimization problem can be regarded as a point on the n-dimensional search space, called particle, and it is assumed that it has no volume and quality. All particles have a fitness value determined by the objective function and a velocity that determines their position and direction of flight. Then the particles can follow the current optimal particle at this rate to search in the solution space, where the particle’s flight speed dynamic adjustments are based on individual flight experience and group flight experience. When the algorithm starts, randomly initialize the population \( Z = (Z_1, Z_2, \ldots, Z_m) \) of \( m \) particles in the feasible solution space. The position of each particle is \( Z_j = (z_{j1}, z_{j2}, \ldots, z_{jn}) \) representing a solution to the problem and calculating the fitness value of each particle based on the objective function. Then perform an iterative search and constantly adjust your position to update the solution. In each iteration, the particles are based on two extreme values \( p_{best} \) and \( g_{best} \). They are respectively the optimal solution searched by the particle itself and the optimal solution of the population group. When the two optimal solutions are found, the optimal particle position is found. The optimal solution to the optimization problem is also found. The standard PSO algorithm flow is as follows.

1. Initialize a group of particles with a population size of \( M \). It includes random position and velocity, local optimal solution and global optimal solution.
2. Adjust particle velocity and position based on speed update formula and position update formula.
3. Evaluate the fitness of each particle.
4. Each particle is compared to its best position \( p_{best} \). If it is better, use it as the current best location \( p_{best} \).
5. Each particle is compared to its best position \( g_{best} \). If it is better, use it as the current best location \( g_{best} \).
6. If the end condition is not reached, go back to the second step (2) and continue with the algorithm.

The iterative termination condition is generally selected as the maximum number of iterations according to the specific problem or the optimal position of the particle swarm search that satisfies the predetermined minimum adaptation threshold, which is usually terminated by the maximum number of iterations.

4.1.2. Supervised selection centre

In the supervised selection centre method, the cluster centre and other parameters are obtained through supervised learning. This method uses errors to correct the learning process, and generally uses the gradient descent method (Bottou, 2012; Burges et al., 2005). The specific algorithm is as follows:

\[
 h_j(x(k)) = \exp(-0.5(x(k) - c_j(k-1))^T \sum_j (k-1) (x(k) - c_j(k-1))) \tag{9}
\]

where \( h_j(x(k)) \) is the hidden layer basis function, \( x(k) = [x_1(k), x_2(k), \ldots, x_n(k)]^T \) is the input of HBFNNs and \( c_j(k-1) \) is hidden layer neuron centre. The output of HBFNNs is as follows:

\[
 y_m(k) = \sum_{j=1}^{m} w_j(k-1) h_j(x(k)) \tag{10}
\]

where \( w_j(k-1) \) is the weight of the neural in the jth hidden layer to the neuron in the output layer at the time instant \( k-1 \) and \( y_m(k) \) is the output of HBFNNs. The network training error is as follows:

\[
 e(k) = y(k) - y_m(k) \tag{11}
\]

where \( y(k) \) is the actual output of the system. The performance index is as follows:

\[
 E(k) = e(k) ^ 2 / 2 \tag{12}
\]
The algorithm of error back propagation is given as follows:

\[
\delta = \frac{\partial E(k)}{\partial y_m(k)} = \frac{\partial E(k)}{\partial e(k)} \frac{\partial e(k)}{\partial y_m(k)} = -e(k) \quad (13)
\]

\[
\frac{\partial E(k)}{\partial w_j(k-1)} = \frac{\partial E(k)}{\partial y_m(k)} \frac{\partial y_m(k)}{\partial w_j(k-1)} = \delta h_j(x(k)) = -e(k) h_j(x(k)) \quad (14)
\]

The weight learning algorithm from the hidden layer to the output layer is as follows:

\[
\Delta w_j(k) = -\mu \frac{\partial E(k)}{w_j(k-1)} = \mu e(k) h_j(x(k)) \quad (15)
\]

\[
w_j(k) = w_j(k-1) + \Delta w_j(k)
+ \alpha (w_j(k-1) - w_j(k-2)) \quad (16)
\]

where \( \mu \) is the learning rate and \( \alpha \) is the momentum factor. The gradient of the hidden layer parameter is as follows:

\[
\frac{\partial E(k)}{\partial \mathbf{c}_j(k-1)} = \frac{\partial E(k)}{\partial h_j(x(k))} \frac{\partial h_j(x(k))}{\partial \mathbf{c}_j(k-1)}
= -e(k) w_j(k-1) h_j(x(k)) \Sigma_j(k-1)
\times (x(k) - \mathbf{c}_j(k-1)) \quad (17)
\]

Then, the learning algorithm of the hidden layer parameter is as follows:

\[
\Delta \mathbf{c}_j(k) = -\mu \frac{\partial E(k)}{\partial \mathbf{c}_j(k-1)} = \mu e(k) w_j(k-1) h_j(x(k))
\times \Sigma_j(k-1) (x(k) - \mathbf{c}_j(k-1)) \quad (18)
\]

\[
\mathbf{c}_j(k) = \mathbf{c}_j(k-1) + \Delta \mathbf{c}_j(k) + \alpha (\mathbf{c}_j(k-1) - \mathbf{c}_j(k-2)) \quad (19)
\]

The gradient of the diagonal matrix \( \Sigma_j(k) \) is as follows:

\[
\frac{\partial E(k)}{\partial \Sigma_j(k-1)} = -e(k) w_j(k-1) h_j(x(k))
\times \Sigma_j(k-1) (x(k) - \mathbf{c}_j(k-1))
\times \frac{\partial (-0.5(x(k) - \mathbf{c}_j(k-1))^T \Sigma_j(k-1)(x(k) - \mathbf{c}_j(k-1)))}{\partial \Sigma_j(k-1)} \quad (20)
\]

where \( \Sigma_{ji}(k-1) \) represents the \( i \)th diagonal element of matrix \( \Sigma_j(k-1), i = 1, 2, ..., n \). Let \( \mathbf{z}_j(k) = x(k) - \mathbf{c}_j(k-1) \). Then we can get

\[
\mathbf{z}_j(k)^T \Sigma_j(k-1) \mathbf{z}_j(k) = \Sigma_{j1}(k-1) z_{j1}(k)^2
+ \Sigma_{j2}(k-1) z_{j2}(k)^2 + \cdots
+ \Sigma_{jn}(k-1) z_{jn}(k)^2 \quad (21)
\]

where \( z_{ji}(k) \) represents the \( i \)th element of matrix \( \mathbf{z}_j(k) \), i.e. \( z_{ji}(k) = x_i(k) - c_{ji}(k-1) \). Substituting (21) into (20) yields

\[
\frac{\partial E(k)}{\partial \Sigma_{ji}(k-1)} = 0.5 e(k) w_j(k-1) h_j(x(k)) z_{ji}(k)^2 \quad (22)
\]

The learning algorithm of the diagonal matrix \( \Sigma_j(k) \) is as follows:

\[
\Delta \Sigma_{ji}(k) = -\mu \frac{\partial E(k)}{\partial \Sigma_{ji}(k-1)}
= -0.5 e(k) w_j(k-1) h_j(x(k)) z_{ji}(k)^2 \quad (23)
\]

\[
\Sigma_{ji}(k) = \Sigma_{ji}(k-1) + \Delta \Sigma_{ji}(k) + \alpha (\Sigma_{ji}(k-1) - \Sigma_{ji}(k-2)) \quad (24)
\]

### 4.1.3. Regularization method

HBFNNs have good fitting ability and high classification accuracy for training data (Hou & Han, 2010). However, due to its extremely high degree of freedom, it is easy to overfit the data. As can be seen from statistical learning theory (Evgeniou, Pontil, & Poggio, 2000; Vapnik & Chervonenkis, 2015), such poor ability of overfitting and generalization can be optimized by adding penalty terms in regularization. Regularization (Bertero, 1986) can solve the overfitting. This is done by minimizing the penalty objective function. When classification is made by machine learning, the SVM method is considered as the most successful classification method due to its high accuracy and fast speed. The classification accuracy of HBFNNs based on regularization optimization can be comparable to SVM (Mahdi & Rouchka, 2011).

### 4.2. Dynamic training

In the dynamic training of HBFNNs, a method of the sequential learning approach is used a lot. Increasing or decreasing the number of hidden neurons can effectively influence the training speed and the convergence time, thus improving the accuracy and dynamic performance (Vuković & Miljković, 2013). The existing dynamic training methods mainly use sequential learning methods, such as resource allocation network (RAN), minimum resource allocation network (MRAN), and growing and pruning (GAP) sequential learning algorithm. The network structure of HBFNNs affects the performance of the entire neural network. When the number of neurons is too small, the accuracy is low. When the number of neurons is too big, the convergence time is too long, and the network training time also increases. Therefore, in order to improve the performance of an HBFNN, it is especially important to effectively cut the number of neurons in the hidden layer of the neural networks, and obtain an appropriate neural network structure, thereby improving the performance of the HBFNN.
4.2.1. **RAN**
The main idea of RAN (Yingwei, Sundararajan, & Saratchandran, 1996, 1998) is to add new hidden nodes to reduce errors when the neural network finds new input samples to make the output error larger during the learning process. The character of this algorithm is online learning. Through this algorithm, a compact network structure with fast convergence speed can be obtained. The RAN algorithm mainly includes adding new hidden layer nodes and adjusting network parameters. In the process of increasing the number of hidden layer nodes, a class of criteria is designed to decide whether to add a new hidden layer node, which include the distance criterion and the error criterion. The distance criterion is that the input sample distance closest to the centre of the hidden node is greater than a certain value. The error criterion is that the difference between the actual output of the neural network and the expected output is greater than a certain value. When the two criteria are satisfied at the same time, a new hidden layer node is allocated. The input sample is taken as the central value of the hidden layer node. The difference between the actual output of the current neural network and the expected output of the sample is taken as the weight of the new hidden layer node. The RAN algorithm uses the gradient descent method to train parameters. The RAN algorithm has some shortcomings, which are as follows:

1. The network parameters in the training process are adjusted by the gradient descent algorithm, so the network convergence speed is slow.
2. The noises and anomalies in the input samples have a large impact on the increase of hidden layer units, so the generalization ability of the network is poor.
3. Due to the influence of noises, the network will add some redundant hidden layer nodes in the learning process, which greatly increases the computational complexity, slows down the network convergence speed and also causes over-fitting of the trained network.

4.2.2. **MRAN**
MRAN is an improved algorithm of RAN. The RAN algorithm can only add the number of neurons but can not remove redundant neurons. So adding a deletion rule can make the network structure more compact (Leong, Saratchandran, & Sundararajan, 2002; Wu et al., 2018). In addition, in order to eliminate the disadvantage that the gradient descent method converges slowly when the parameter is updated, the MRAN algorithm uses the EKF algorithm to update parameters (Huang, Saratchandran, & Sundararajan, 2005). On the basis of RAN, the hidden layer node deletion criterion is added. This is the main difference between the MRAN algorithm and the RAN algorithm.

4.2.3. **GAP**
We know that the key to determine the structure of HBFNNs is the number of hidden layer nodes and the selection of centres. In the process of network training, the number of hidden layer nodes, the acquisition and adjustment of centres are closely related to the input samples. In addition, the number of repeated training of input samples directly affects the real-time learning ability of the networks. It can be found that both RAN and MRAN algorithms need to save previously learned samples in the learning process. When a new sample input, the addition criterion and the deletion criterion of the hidden layer nodes need to be judged according to the previous sample information. This makes RAN and MRAN not purely sequential local incremental learning algorithms.

In order to improve the convergence speed and the algorithm performance, Huang, Saratchandran, and Sundararajan (2004); Zhang, Huang, Sundararajan, and Saratchandran (2007) proposed the GAP sequential learning algorithm, and applied it for the training of RBFNNs. The GAP algorithm can directly simulate complex nonlinear mappings using input and output data with a simple topology without the need to repeat learning inputs. However, it also has certain requirements on the probability density distribution of input samples, and different probability density distributions which have different structures. Vukovi and Miljkovi (2013) used the gaussian mixture probability model (GMM) to optimize the GAP algorithm and used the optimized algorithm to train the HBFNNs. The GMM can simplify the calculation of the algorithm without distinguishing the probability density distribution of different input samples.

5. Applications of HBFNNs
In the past 10 years, great progress has been made in both the theory and application of HBFNNs. Many practical problems are difficult to be solved by other machine learning methods that have been dealt successfully by HBFNNs in the fields of pattern recognition, intelligent robot, automatic control, prediction and estimation, biology, medicine and economy. Some common application scenarios are summarized as follows.

5.1. **HBF neural networks observer**
Now the development direction of intelligent control is developing in the direction of safety, efficiency, fault tolerance and self-adaptation, and is combined with the
design of the safety system. In the process of implementing active autonomous systems, neural networks’ observer-based methods play a crucial role in the processing of nonlinear systems (Wang, Jiang, & Liu, 2016). With the rapid development of related technology fields over time, some calculation-based algorithms have evolved. Observer technology is becoming more and more mature. Tracking and observing system status can be fast, accurate and self-learning. The addition of various intelligent algorithms has made observer-based methods more and more dominant in dealing with nonlinear problems. There are many methods based on observers, but none of them is perfectly able to track the system perfectly. However, the neural network-based observer has a good approximation effect in the nonlinear part, and has a good tracking effect on the output of the whole system. The neural networks observer can overcome the fluctuation of the system, and quickly and accurately keeps up with the system and reacts. The method of neural networks observer is paid more and more attention. HBFNNs are increasingly used in observer design because of its short training time and high precision. For example, the HBFNNs observer is used for fault diagnosis in Wen et al. (2015). With the development of modern intelligent technology, a large class of intelligent algorithms has emerged. For example, particle swarm optimization, genetic algorithms, artificial immune algorithms, etc., followed by bio-population algorithms, such as ant colony algorithm and artificial fish swarm algorithm. Using these intelligent optimization algorithms to train HBFNN can show very good performance in data processing and global optimization. Therefore, the observer method combining intelligent algorithm and HBFNN will play an important role in future research.

5.2. Networked control system based on HBF neural networks prediction

Networked control systems have many advantages such as low cost, high reliability and flexible structure (Gupta & Chow, 2009; Yang, 2006) and it has been widely used in telemedicine, robotics, aerospace, etc. However random packet loss and delay often occur in networked control systems due to the introduction of the networks, so overcoming this problem has become a hot topic in control engineering field. The network constraint problems not only reduce the performance of the system, and even worse destroy system stability. Therefore, the study of the stability, fault detection, state estimation (Du & Du, 2009; Hua, Yu, & Guan, 2014) and prediction of output data has received more and more attention (Gupta & Chow, 2009).

The HBFNNs predictive controller can solve the fault detection problem of the networked control system with sensor data loss. The HBFNNs predictive controller can adjust learning efficiency online and predict output values more accurately. When a packet loss occurs, the system output data transmitted from the sensor cannot be normally transmitted to the controller through the network, causing the system to fail to operate normally. At this time, the predictive controller starts to function. According to the output information of the lost packet value of the system, the neural networks predictive method is used to estimate the output data information at the current moment. This method effectively offsets the adverse effects brought by the packet loss to the system. In addition, in order to analyse the stability of the system, the fault observer is designed according to the system packet loss situation. When the system is running normally, the observer is stable. When the system fails, it can detect the fault according to the observer.

In order to solve the adverse effects brought by the delay to the system, Na, Ren, Shang, and Guo (2012) propose an adaptive neural networks control method for nonlinear feedback systems with input delay. This method combines the adaptive prediction period of a high-order neural network observer to obtain predictions of future system states. This method can be used to overcome the problem of input delay and nonlinearity in control design. Simulation experiments show that the neural network predictive controller method can effectively predict the output data of the system, thus making up for the adverse effects of delay on the system.

5.3. Application in the field of face recognition

With the continuous development of pattern recognition, biometric identification has been increasingly used in the field of identity verification. Face recognition (Jafri & Arabnia, 2009), as a pattern recognition technology, has important applications in various fields due to its easy-to-identify objects, user-friendliness and low intrusion. HBFNN has been shown to have good approximation capabilities for any nonlinear function. The face recognition method based on the HBF neural network has the advantages of fast recognition speed and high accuracy, and has been successfully applied in the field of face recognition. Deng, Jin, Zhen, and Huang (2005) proposed a method combining PCA and LDA and used the nearest neighbour discriminant method for classification and recognition. After the training images are extracted by PCA and LDA, the feature subspace of the training samples is obtained. Then it is normalized and processed and used as an input to the neural network and train the neural network. The literature (Sahoolizadeh, Heidari, & Dehghani, 2008; Toygar & Adnan, 2003) performs the
same processing on the face image on the test set for classification and recognition.

5.4. Other fields

The HBFNNs are also widely used in the public health field. Neural networks are mostly used for the detection and diagnosis of the prevalence of certain diseases, health statistics and so on (Nieminen et al., 2003; Vicente et al., 2012). Cancer data are classified using regularized HBFNNs to determine cancer types. Cancer classification is the aggregation of human genes into a sample so that it can be correctly classified. When using HBFNNs for data classification, Mahdi and Rouchka (2009) used local dimensionality reduction and added penalty functions. This method improves the classification accuracy. There is an increase in the classification accuracy from 92.3% to 96.4% without increasing the number of penalty items.

The HBFNNs are also applied in the economic field (Kimoto et al., 1990). For example, the application of the market price forecast field. The analysis of market price forecasts for changes in commodity prices can be attributed to a comprehensive analysis of many factors that influence market supply and demand. The traditional statistical economic method is difficult to make scientific predictions on price changes because of its inherent limitations. HBFNNs have the characteristics of high degree of freedom and variable parameters, and has good performance when dealing with incomplete, fuzzy uncertain or irregular data. Therefore, the use of HBFNNs for price forecasting has advantages that traditional methods cannot match. Starting from the market price determination mechanism, based on complex factors such as household disposable income, loan interest rate and urbanization level affecting commodity prices, a more accurate and reliable model can be established through the HBF neural network. The model can scientifically predict the trend of changes in commodity prices and obtain accurate evaluation results.

HBFNNs also are used for risk assessment (Yu, Wang, & Lai, 2008). Risk refers to the possibility of economic or financial losses, damage caused by natural damage or uncertainty in the course of engaging in a particular activity. The best way to prevent risk is to make scientific predictions and assessments of risks in advance. The prediction idea of applying HBFNNs is mainly divided into two steps. First, construct the structure and algorithm of the credit risk model suitable for the actual situation based on the actual risk source. The second obtains the risk assessment coefficient and determines the solution to the actual problem. The empirical analysis using this model can make up for the lack of subjective assessment and achieve satisfactory results. The role of HBF neural networks in the field of transportation is also becoming more and more important. In recent years people have begun to study the application of neural networks in transportation systems (Srinivasan, Choy, & Cheu, 2006). The problem of transportation is highly non-linear, and the available data are usually large and complex. The use of neural networks to deal with related problems has its enormous advantages. Applications include simulation of vehicle driver behaviour, parameter estimation, road maintenance, vehicle detection and classification, traffic pattern analysis, cargo operations management, traffic flow forecasting, transportation strategy and economy, traffic environmental protection, air transportation, automatic navigation of ships and the identification of ships, subway operations, traffic control. All these applications in different field have experienced good results.

HBFNNs can be used to eliminate nonlinear interference. Sergiy and Andrzej (2002) proposed a nonlinear interference elimination filter based on HBFNNs. Due to the flexible approximate characteristics of HBFNNs, it was able to achieve better results for the elimination of nonlinear interference. Finally, the simulation comparison with the filter made by RBF networks verifies the superiority of HBFNNs in eliminating system interference. In addition, modern control applications are complex nonlinear environments and require high real-time performance. Therefore, real-time processing of nonlinear systems and control objects is particularly important in research and practical operations. Witkowski, Neumann, and Ruckert (1999) proposed a hardware implementation method of HBF digital neural networks for function approximation. The learning and memory of neural networks are realized on the hardware, and the high performance networks’ computing is realized. This hardware has the function of real-time function approximation, which opens the door to the use of function approximation for the real-time learning requirements of online learning. Neural networks also have applications in sensor modelling (Menacer, Kadr, & Dibi, 2018), it makes the application of HBFNNs in this field a possibility. In the recent rise of natural language processing (NLP), neural networks have also demonstrated superior performance (Chen, Tian, & Liu, 2018; Zhao, Feng, Wu, & Yan, 2017).

6. Future trends of HBFNNs

HBFNN has the advantage of high degree of freedom, eliminating interference between data, simple network structure and fewer network parameters to be trained. This makes HBFNNs to overcome the shortcomings of traditional feedforward neural networks when dealing with complex nonlinear systems with slow training speeds and large correlations between data. It greatly improves the
application performance of HBFNN in the field of expert systems, intelligent control and combinatorial optimization. In addition, the combination of HBFNN and other intelligent optimization algorithms will drive the development of neural networks and information processing technologies. The application of information geometry in HBFNNs research has opened up a new way for the theoretical study of HBFNNs. The research of neural computers is developing rapidly, and the optoelectronic combination computer provides good conditions for the development of HBFNNs. In recent years, the development of hot deep learning algorithms and artificial intelligence technology has provided a new development direction for HBFNNs. In the future, HBF neural networks are likely to move toward multi-layer, multi-algorithm, and multi-structure.

7. Conclusions

The paper briefly introduces HBFNNs, which mainly includes the development process, training algorithm, practical application and development trends. Compared with other neural networks, both theory and application have developed rapidly because it can solve complex nonlinear problems efficiently under the background of limited resources and high requirements for accuracy and rapidity. First, the paper introduces the six stages of neural networks development and the basic knowledge of HBFNNs in detail as well as RBFNNs. Then the training algorithm is summarized as the main content of this paper. Due to the advantages of the algorithm, HBF is widely used in many fields which are listed in the article clearly. Finally, the development trends are pointed out based on the theory and technology at the current stage.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by the National Natural Science Foundation of China under Grant 61673023, the Beijing Natural Science Foundation under Grant 4182019, the Basic Scientific Research Foundation of Beijing Municipal Education Commission under Grant 110052971921/026, and the NCUT Yujie Talent Training Program under Grant 107051360019XT133/001.

References

Bertero, M. (1986). Regularization methods for linear inverse problems. In G. Talenti (Ed.), Inverse problems (pp. 52–112). Berlin: Springer.

Bottou, L. (2012). Stochastic gradient descent tricks. In G. Montavon, G. B. Orr, & K. R. Mueller (Eds.), Neural networks: Tricks of the trade (pp. 421–436). Berlin: Springer.

Bradley, P. S., & Fayyad, U. M. (1998, July). Refining initial points for K-means clustering. Paper presented at the ICML (Vol. 98, pp. 91–99). San Francisco: Morgan Kaufmann.

Broomhead, D. S., & Lowe, D. (1988). Radial basis functions, multi-variable functional interpolation and adaptive networks (No. RSRE-MEMO-4148). Malvern, United Kingdom: Royal Signals and Radar Establishment.

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., & Hullender, G. N. (2005). Learning to rank using gradient descent. Paper presented at the Proceedings of the 22nd international conference on machine learning (ICML-05), Bonn, Germany (pp. 89–96).

Chen, C. P., & Liu, Z. (2017, May). Broad learning system: A new learning paradigm and system without going deep. Paper presented at the 2017 32nd youth academic annual conference of Chinese Association of Automation (YAC) (pp. 1271–1276). Silver Spring, MD: IEEE.

Chen, H. Z., Tian, G. H., & Liu, G. L. (2018). A selective attention guided initiative semantic cognition algorithm for service robot. International Journal of Automation and Computing, 15(5), 559–569.

Chen, H., Wang, J. H., & Wang, X. Z. (2003, November). A new similarity measure based on feature weight learning. Paper presented at the Proceedings of the 2003 international conference on machine learning and cybernetics (IEEE Cat. No. 03EX693) (Vol. 1, pp. 33–36). Silver Spring, MD: IEEE.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine Learning, 20(3), 273–297.

Dash, C. S. K., Behera, A. K., Dehuri, S., & Cho, S. B. (2016). Radial basis function neural networks: A topical state-of-the-art survey. Open Computer Science, 6(1), 33–63.

Deng, H. B., Jin, L. W., Zhen, L. X., & Huang, J. C. (2005). A new facial expression recognition method based on local Gabor filter bank and PCA plus LDA. International Journal of Information Technology, 11(11), 86–96.

Ding, C., & He, X. (2004, July). K-means clustering via principal component analysis. Paper presented at the Proceedings of the twenty-first international conference on machine learning (p. 29). New York: ACM.

Du, F., & Du, W. (2009, July). RBFNNs control and novel smith predictor for networked control systems. Paper presented at the 2009 Asia-pacific conference on information processing (Vol. 1, pp. 67–70). Silver Spring, MD: IEEE.

Eberhart, R., & Kennedy, J. (1995, November). Particle swarm optimization. Paper presented at the Proceedings of the IEEE international conference on neural networks, Perth, Australia (Vol. 4, pp. 1942–1948).

Evgeniou, T., Pontil, M., & Poggio, T. (2000). Regularization networks and support vector machines. Advances in Computational Mathematics, 13(1), 1.

Gupta, R. A., & Chow, M. Y. (2009). Networked control system: Overview and research trends. IEEE Transactions on Industrial Electronics, 57(7), 2527–2535.

Ha, V. K., Ren, J., Xu, X., Zhao, S., Xie, G., & Vargas, V. M. (2018, July). Deep learning based single image super-resolution: A survey. Paper presented at the international conference on brain inspired cognitive systems (pp. 106–119). Cham: Springer.

Hebb, D. O. (1952). The organisation of behaviour: A neuropsychological theory. New York: Wiley.

Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. Neural Computation, 18(7), 1527–1554.
Hou, M., & Han, X. (2010). Constructive approximation to multivariate function by decay RBFNNs. *IEEE Transactions on Neural Networks*, 21(9), 1517–1523.

Hua, C., Yu, C., & Guan, X. (2014). Neural networks observer-based networked control for a class of nonlinear systems. *Neurocomputing*, 133, 103–110.

Huang, G. B., Saratchandran, P., & Sundararajan, N. (2004). An efficient sequential learning algorithm for growing and pruning RBF (GAP-RBF) networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(6), 2284–2292.

Huang, G. B., Saratchandran, P., & Sundararajan, N. (2005). A generalized growing and pruning RBF (GGAP-RBF) neural networks for function approximation. *IEEE Transactions on Neural Networks*, 16(1), 57–67.

Jafri, R., & Arabnia, H. R. (2009). A survey of face recognition techniques. *Jips*, 5(2), 41–68.

Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R., & Wu, A. Y. (2002). An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7), 881–892.

Karayiannis, N. B., & Mi, G. W. (1997). Growing radial basis neural networks: Merging supervised and unsupervised learning with networks growth techniques. *IEEE Transactions on Neural Networks*, 8(6), 1492–1506.

Kimoto, T., Asakawa, K., Yoda, M., & Takeoka, M. (1990, June). Stock market prediction system with modular neural networks. Paper presented at the 1990 IJCNN international joint conference on neural networks (pp. 1–6). Silver Spring, MD: IEEE.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). *Imagenet classification with deep convolutional neural networks*. Paper presented at the Advances in neural information processing systems, Lake Tahoe, NV (pp. 1097–1105).

LeCun, Y. (1989). Generalization and networks design strategies. In *Connectionism in perspective* (Vol. 19). Amsterdam: Elsevier.

Leong, T. K., Saratchandran, P., & Sundararajan, N. (2002). Real-time performance evaluation of the minimal radial basis function networks for identification of time varying nonlinear systems. *Computers & Electrical Engineering*, 28(2), 103–117.

Likas, A., Vlassis, N., & Verbeek, J. J. (2003). The global k-means clustering algorithm. *Pattern Recognition*, 36(2), 451–461.

Lu, Y. Z. (2008). A novel face recognition algorithm for distinguishing faces with various angles. *International Journal of Automation and Computing*, 5(2), 193–197.

Mahdi, R. N., & Rouchka, E. C. (2009, December). Feature selection in cancer classification from mRNA data based on localized dimension reduction. Paper presented at the 2009 International conference on machine learning and applications (pp. 443–448). Silver Spring, MD: IEEE.

Mahdi, R. N., & Rouchka, E. C. (2011). Reduced HyperBF networks: Regularization by explicit complexity reduction and scaled Rprop-based training. *IEEE Transactions on Neural Networks*, 22(5), 673–686.

Mao, J., & Jain, A. K. (1996). A self-organizing networks for hyper-ellipsoidal clustering (HEC). *IEEE Transactions on Neural Networks*, 7(1), 16–29.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4), 115–133.

Menacer, F., Kadr, A., & Dibi, Z. (2018). Modeling of a smart nano force sensor using finite elements and neural networks. *International Journal of Automation and Computing*. doi:10.1007/s11633-018-1155-6

Minsky, M., & Papert, S. (1969). *An introduction to computational geometry*. Cambridge: MIT Press.

Na, J., Ren, X., Shang, C., & Guo, Y. (2012). Adaptive neural networks predictive control for nonlinear pure feedback systems with input delay. *Journal of Process Control*, 22(1), 194–206.

Nieminen, P., Hakama, M., Viikki, M., Tarkkanen, J., & Anttila, A. (2003). Prospective and randomised public-health trial on neural networks assisted screening for cervical cancer in Finland: Results of the first year. *International Journal of Cancer*, 103(3), 422–426.

Poggio, T., & Girosi, F. (1990). Networks for approximation and learning. *Proceedings of the IEEE*, 78(9), 1481–1497.

Poggio, T., Mhaskar, H., Rosasco, L., Miranda, B., & Liao, Q. (2017). Why and when can deep-but not shallow-networks avoid the curse of dimensionality: A review. *International Journal of Automation and Computing*, 14(5), 503–519.

Rao, A. M., Ramji, K., Rao, B. S. S., Vasu, V., & Poneeth, C. (2017). Navigation of non-holonomic mobile robot using neuro-fuzzy logic with integrated safe boundary algorithm. *International Journal of Automation and Computing*, 14(3), 285–294.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6), 386–408.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1988). Learning representations by back-propagating errors. *Cognitive Modeling*, 5(3), 1.

Sahoolizadeh, A. H., Heidari, B. Z., & Deghhani, C. H. (2008). A new face recognition method using PCA, LDA and neural networks. *International Journal of Computer Science and Engineering*, 2(4), 218–223.

Schwenker, F., Kestler, H. A., & Palm, G. (2001). Three learning phases for radial-basis-function networks. *Neural Networks*, 14(4–5), 439–458.

Sergiy, A. V., & Andrzejc, C. (2002). Hyper radial basis function neural networks for interference cancellation with nonlinear processing of reference signal. *Digital Signal Processing*, 11(3), 204–221.

Srinivasan, D., Choy, M. C., & Cheu, R. L. (2006). Neural networks for real-time traffic signal control. *IEEE Transactions on Intelligent Transportation Systems*, 7(3), 261–272.

Su, M. C., & Liu, T. K. (2001). Application of neural networks using quadratic junctions in cluster analysis. *Neurocomputing*, 37(1–4), 165–175.

Svozil, D., Kvasnicka, V., & Pospichal, J. (1997). Introduction to multi-layer feed-forward neural networks. *Chemometrics and Intelligent Laboratory Systems*, 39(1), 43–62.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., e. a. & Rabinovich, A. (2015). *Going deeper with convolutions*. Paper presented at the Proceedings of the IEEE conference on computer vision and pattern recognition, Boston, MA (pp. 1–9).

Toygar, Ö, & Adnan, A. (2003). Face recognition using PCA, LDA and ICA approaches on colored images. *Istanbul University-Journal of Electrical & Electronics Engineering*, 3(1), 735–743.

Vapnik, V. N., & Chervonenkis, A. Y. (2015). On the uniform convergence of relative frequencies of events to their probabilities. In V. Vovk, H. Papadopoulos, & A. Gammerman (Eds.), *Measures of complexity* (pp. 11–30). Cham: Springer.
Vicente, H., Dias, S., Fernandes, A., Abelha, A., Machado, J., & Neves, J. (2012). Prediction of the quality of public water supply using artificial neural networks. *Journal of Water Supply: Research and Technology-AQUA*, 61(7), 446–459.

Vuković, N., & Miljković, Z. (2013). A growing and pruning sequential learning algorithm of hyper basis function neural networks for function approximation. *Neural Networks*, 46, 210–226.

Wang, R., Jiang, B., & Liu, J. (2016, August). Fault detection and estimation for a class of nonlinear systems based on neural networks observer. Paper presented at the 2016 IEEE Chinese guidance, navigation and control conference (CGNCC) (pp. 1216–1219). Silver Spring, MD: IEEE.

Wen, X., Zhang, X., & Zhu, Y. (2015). Design of fault detection observer based on hyper basis function. *Tsinghua Science and Technology*, 20(2), 200–204.

Werbos, P. J. (1974). *New tools for prediction and analysis in the behavioral sciences* (Ph.D. dissertation). Harvard University.

Witkowski, U., Neumann, T., & Ruckert, U. (1999, April). Digital hardware realization of a hyper basis function networks for online learning. Paper presented at the Proceedings of the seventh international conference on microelectronics for neural, fuzzy and bio-inspired systems (pp. 205–211). Silver Spring, MD: IEEE.

Wu, C., Kong, X. R., & Yang, Z. G. (2018, May). An online self-adaptation learning algorithm for hyper basis function neural network. Paper presented at the 2018 2nd IEEE advanced information management, communicates, electronic and automation control conference (IMCEC) (pp. 215–220). Silver Spring, MD: IEEE.

Yang, T. C. (2006). Networked control system: A brief survey. *IEEE Proceedings-Control Theory and Applications*, 153(4), 403–412.

Yang, H. J., & Tan, M. (2018). Sliding mode control for flexible-link manipulators based on adaptive neural networks. *International Journal of Automation and Computing*, 15(2), 239–248.

Yingwei, L., Sundararajan, N., & Saratchandran, P. (1996, May). Adaptive nonlinear system identification using minimal radial basis function neural networks. Paper presented at the 1996 IEEE international conference on acoustics, speech, and signal processing conference proceedings (Vol. 6, pp. 3521–3524). Silver Spring, MD: IEEE.

Zhang, X. B., & Jiang, L. (2009, August). A new validity index of fuzzy c-means clustering. Paper presented at the 2009 International conference on intelligent human–machines systems and cybernetics (Vol. 2, pp. 218–221). Silver Spring, MD: IEEE.

Zhao, B., Feng, J., Wu, X., & Yan, S. (2017). A survey on deep learning-based fine-grained object classification and semantic segmentation. *International Journal of Automation and Computing*, 14(2), 119–135.

Zhou, Y., & Jiang, J. (2015, December). An FPGA-based accelerator implementation for deep convolutional neural networks. Paper presented at the 2015 4th international conference on computer science and network technology (ICCSNT) (Vol. 1, pp. 829–832). Silver Spring, MD: IEEE.