Available at
www.ElsevierMathematics.com
POWERED BY SCIENCE @ DIRECT®

ELSEVIER Computers and Mathematics with Applications 46 (2003) 1861–1874

An International Journal
computers & mathematics with applications

www.elsevier.com/locate/camwa

Interpolation Functions of Feedforward Neural Networks

HONG-XING LI
Department of Automation
Tsinghua University
Beijing, 100084, China

E. S. LEE*
Department of Industrial and Manufacturing Systems Engineering
Kansas State University, Manhattan, KS 66506 U.S.A.
eslee@ksu.edu

(Received and accepted March 2002)

Abstract—Mathematical essence and structures of the feedforward neural networks are investigated in this paper. The interpolation mechanisms of the feedforward neural networks are explored. For example, the well-known result, namely, that a neural network is an universal approximator, can be concluded naturally from the interpolative representations. Finally, the learning algorithms of the feedforward neural networks are discussed. © 2003 Elsevier Ltd. All rights reserved.

Keywords—Neural network, Fuzzy neural network, Interpolation function, Mathematical neuron, Mathematical neural network.

1. INTRODUCTION

Because of its learning and representation abilities, neural networks have been applied to various practical areas. In this paper, we wish to study the rich meanings of the neural approach from the mathematical and interpolation standpoint. One of the problems is the transfer function generally used in the literature, where there exists no paralysis or inhibition after excitation. Although, in reality, a neuron cannot stay excited indefinitely. In this work, we assume there exists paralysis. Based on this assumption, we obtained trapezoidal and triangular neurons. By the use of these neurons, we can obtain various interpolation functions, which show naturally why the neural network is an universal approximator.

In the following, mathematical neural network and fuzzy neurons are first defined in the next section. Then, in the following two sections, we obtained various interpolation functions. Finally, the learning algorithm is discussed.

This work is partially supported by the National Natural Science Foundation of China.
*Author to whom all correspondence should be addressed.

2. MATHEMATICAL NEURON AND MATHEMATICAL NEURAL NETWORK

The most frequently encountered artificial neuron models are neurons with multiple inputs and single output such as the McCulloch and Pitts (MP) neuron shown in Figure 1, where $X = (x_1, x_2, \dots, x_n)$ represents the input vector, $W = (w_1, w_2, \dots, w_n)^{\mathrm{T}}$ a weight vector, θ the threshold value, and φ the input-output function or the activation function of the neuron.

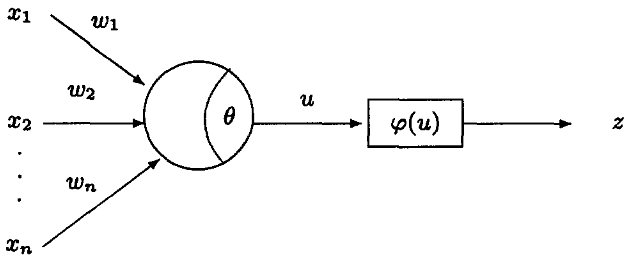

Figure 1. McCulloch and Pitts artificial neuron.

Let z represent the output, then the relationship between input and output can be represented as

$$z = \varphi\left(\sum_{i=1}^{n} w_i x_i - \theta\right) = \varphi\left(X \cdot W - \theta\right). \tag{1}$$

Depending on the transfer function used, this type of artificial neuron can be further divided into the two-valued output model and the continuous-valued output model.

MP NEURON WITH TWO-VALUED OUTPUT. For the two-valued output, φ is a step function

$$\varphi(u) = \begin{cases} 1, & u \geq 0, \\ 0, & u < 0. \end{cases} \tag{2}$$

MP NEURON WITH CONTINUOUS-VALUED OUTPUT. Here the transfer function, φ, can be a trapezoidal function, a sigmoidal function, or a displacement function

$$\varphi(u) = \begin{cases} 0, & u < u_0, \\ \dfrac{u - u_0}{u_1 - u_0}, & u_0 \leq u \leq u_1, \\ 1, & u > u_1, \end{cases} \tag{3}$$

$$\varphi(u) = \left[1 + \exp(-u + c)\right]^{-1}, \qquad c = \text{constant}, \tag{4}$$

$$\varphi(u) = u + c, \qquad c = \text{constant}. \tag{5}$$

Notice that, when $c = 0$, the displacement function reduces to an identity function.

REMARK 1. When $(\forall u)$ $(\varphi(u) \in [0,1])$, the neuron can be called a fuzzy neuron. Furthermore, the neural network linked by fuzzy neurons should be called a fuzzy neural network. This definition of a fuzzy neuron is weak and it can be strengthened. Notice the term "$X \cdot W$" in equation (1) which means $\sum_{i=1}^{n} w_i x_i$, if the pair of operators $(+, \cdot)$ in this term is replaced by the pair of operators $(\vee, \wedge)$, then this term becomes $\vee_{n=1}{}^{n}(w_i \wedge x_i)$. With this new term, the neuron becomes a strengthened fuzzy neuron.

Based on the above-defined neuron models with multiple input and single output, we can define a "mathematical neuron" as follows.

DEFINITION 1. *Let R be the real number field, for any $D \subset R^n$ and $E \subset R$, a function,*

$$f : D \longrightarrow E, \qquad (x_1, x_2, \dots, x_n) \longmapsto z = f(x_1, x_2, \dots, x_n),$$

is called a mathematical neuron. When $f(D) = \{0,1\}$ or $\{-1,1\}$, f is called a two-valued mathematical neuron; when $f(D) \subset [0,1]$, f is called a fuzzy mathematical neuron.

Clearly, the neuron defined by equation (1) is a special example of the mathematical neuron

$$z = f(x_1, x_2, \dots, x_n) \triangleq \varphi\left(\sum_{i=1}^{n} w_i x_i - \theta\right).$$

A network is called a fuzzy mathematical neural network, if it is formed by fuzzy mathematical neurons.

Now, let us consider the structure of the activation function. Equation (2) can be generalized and rewritten as

$$\varphi'(u) = \begin{cases} 1, & u \geq \theta, \\ 0, & u < \theta. \end{cases} \tag{6}$$

Thus, equation (1) simplified to

$$z = \varphi'\left(\sum_{i=1}^{n} w_i x_i\right) = \varphi'(W \cdot X), \tag{7}$$

then $\varphi'(W \cdot X) = \varphi(W \cdot X - \theta)$. In other words, φ and φ' are equivalent. The threshold θ has been absorbed into φ'. Let $y = W \cdot X - \theta$, then $\varphi'(y)$ has the form

$$\varphi'(y) = \begin{cases} 1, & y \geq \theta, \\ 0, & y < \theta. \end{cases} \tag{8}$$

Moreover, the activation functions defined by equations (3)–(5) can be treated similarly.

If equation (2) is rewritten as

$$\varphi(u) = \begin{cases} 1, & u \geq \theta, \\ 0, & u < \theta, \end{cases} \tag{9}$$

then the relationship between input and output represented by equation (1) can be simplified as

$$z = \varphi\left(\sum_{i=1}^{n} w_i x_i\right) = \varphi(W \cdot X). \tag{10}$$

Figure 2 illustrates the activation function defined by equation (9).

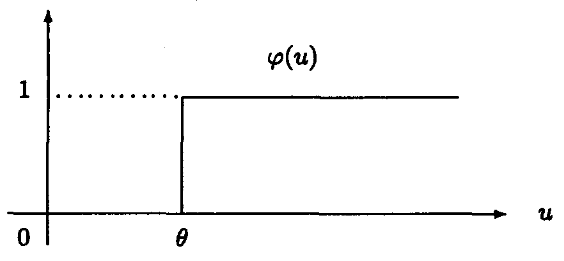

Figure 2. The activation function.

From Figure 2, we see that the neuron will be excited when $u \geq \theta$, i.e., $\varphi(u) = 1$. However, there is a problem in this plot, because there is no limit on the value of u. The question is whether there exists an upper bound on u such that $\varphi(u) = 0$ when $u \geq u_0$.

In reality, for any neuron, u cannot increase indefinitely. From a biological standpoint, once the neuron is excited, there should exist a bound $\theta' > \theta$ such that when $u \geq \theta'$, the neuron will turn to paralysis or inhibition. In other words, there should exist an $\theta' > \theta$ such that $\varphi(u) = 0$ when $u \geq \theta'$. Thus, equation (9) should be rewritten as

$$\varphi(u) = \begin{cases} 1, & \theta \leq u \leq \theta', \\ 0, & \text{otherwise}. \end{cases} \tag{11}$$

Figure 3 illustrates an activation function with bounded excitation as defined by equation (11). In fact, excitation and inhibition should not happen suddenly or catastrophically. There should exist transition or buffer zones for both excitation and inhibition. Thus, a more reasonable activation function may be the following trapezoid, which is illustrated in Figure 4.

$$\varphi(u) = \begin{cases} \dfrac{u-\theta_1}{\theta_2-\theta_1}, & \theta_1 \leq u \leq \theta_2, \\ 1, & \theta_2 < u < \theta_3, \\ \dfrac{\theta_4-u}{\theta_4-\theta_3}, & \theta_3 \leq u \leq \theta_4, \\ 0, & \text{otherwise}. \end{cases} \tag{12}$$

Furthermore, in a certain special situation, we may have the equality, $\theta_2 = \theta_3$. Thus, equation (12) may degenerate into the following "triangle wave" activation function (see Figure 5):

$$\varphi(u) = \begin{cases} \dfrac{u-\theta_1}{\theta_2-\theta_1}, & \theta_1 \leq u \leq \theta_2, \\ \dfrac{\theta_3-u}{\theta_3-\theta_2}, & \theta_2 < u \leq \theta_3, \\ 0, & \text{otherwise}. \end{cases} \tag{13}$$

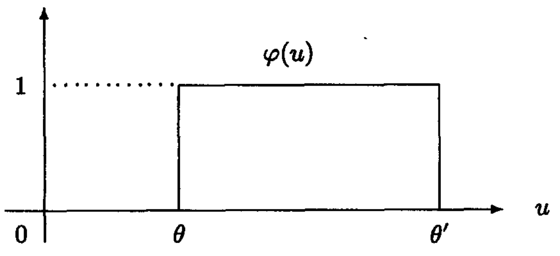

Figure 3. Activation function with upper bound.

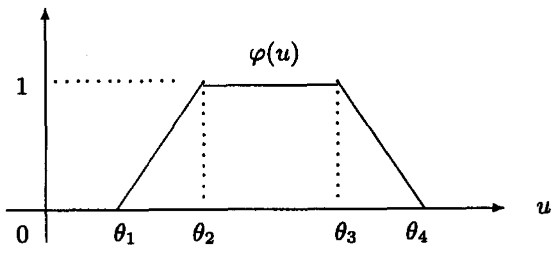

Figure 4. Trapezoidal activation function.

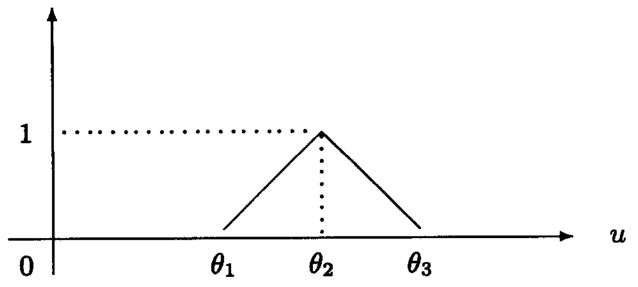

Figure 5. Triangle wave activation function.

For a given n-dimensional vector $W = (w_1, w_2, \dots, w_n)^\top$, if we map $\Sigma : R^n \longrightarrow R$ such that

$$(x_1, x_2, \dots, x_n) \longmapsto \Sigma(x_1, x_2, \dots, x_n) \triangleq \sum_{i=1}^{n} w_i x_i = X \cdot W,$$

then all the neurons defined above can be expressed as $f = \varphi \circ \Sigma$, i.e.,

$$\begin{aligned} z &= f(x_1, x_2, \dots, x_n) = (\varphi \circ \Sigma)(x_1, x_2, \dots, x_n) \\ &= \varphi(\Sigma(x_1, x_2, \dots, x_n)) = \varphi\left(\sum_{i=1}^{n} w_i x_i\right) = \varphi(X \cdot W). \end{aligned} \tag{14}$$

This reflects the well-known fact that neurons have a basic feature of integrating by Σ first and then activating by φ.

3. INTERPOLATION OF FEEDFORWARD NEURAL NETWORK

A typical feedforward neural network is shown in Figure 6. Since a complicated network can always be decomposed into simpler ones, we shall start our investigation of the mathematical essence and structure of neural network from some simpler components.

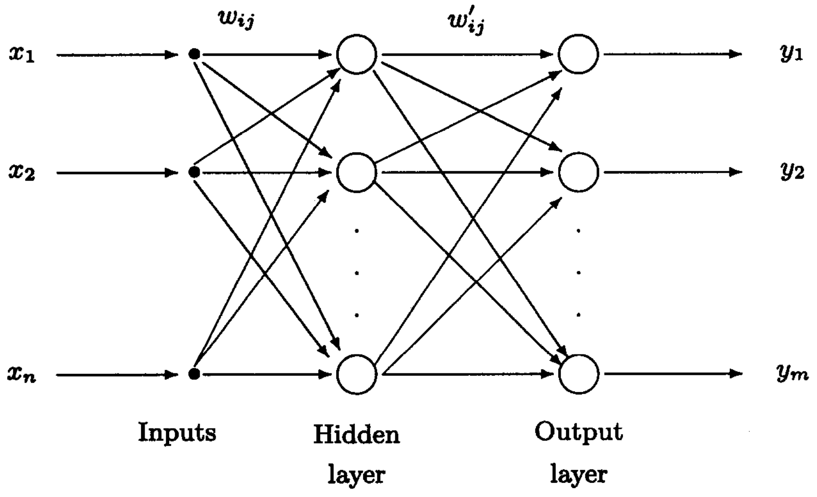

Figure 6. Feedforward neural network.

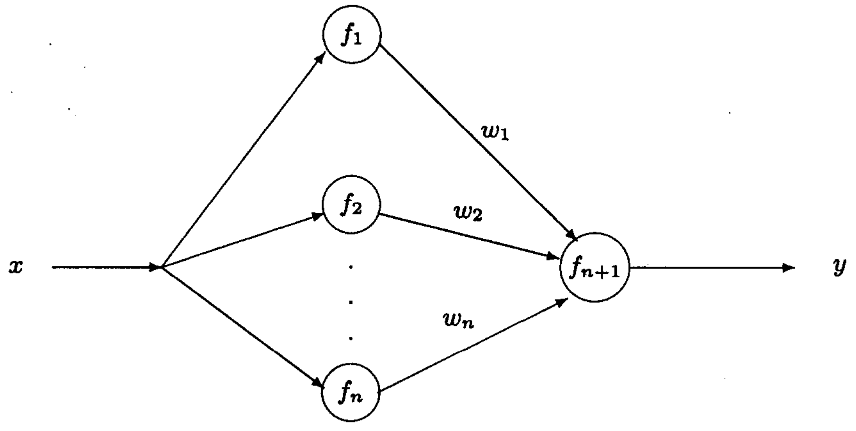

Figure 7. A two-layer feedforward neural network with single input and single output.

In most cases, a feedforward neural network can be regarded as a mapping:

$$F : R^n \longrightarrow R^m, \qquad (x_1, \dots, x_n) \longmapsto F(x_1, \dots, x_n) = (y_1, \dots, y_m).$$

When $n = m = 1$, F reduces to a function of a single variable: $y = F(x)$, which can be represented by a two-layer feedforward neural network with single input and single output as shown Figure 7.

Now, let the activation function of the neuron f_i be φ_i, $i = 1, 2, \dots, n+1$, the connection weight from f_i to f_{n+1} be w_i, $i = 1, 2, \dots, n$, and φ_{n+1} be the identity function: $\varphi_{n+1}(u) = u$; then from equation (10), the output of the network, which is illustrated in Figure 7, can be represented by

$$\begin{aligned} F(x) = y &= f_{n+1}(f_1(x), f_2(x), \dots, f_n(x)) \\ &= f_{n+1}(\varphi_1(x), \varphi_2(x), \dots, \varphi_n(x)) = \varphi_{n+1}\left(\sum_{i=1}^{n} \varphi_i(x) w_i\right) \\ &= \sum_{i=1}^{n} \varphi_i(x) w_i. \end{aligned} \tag{15}$$

Clearly, equation (15) is just the general formula of interpolation, where the activation functions $\varphi_1, \varphi_2, \dots, \varphi_n$ form the basic functions of interpolation. In other words, given an appropriate set of basic functions, $\varphi_1, \varphi_2, \dots, \varphi_n$, equation (15) forms an interpolation function.

EXAMPLE 1. In equation (15), let the activation functions φ_i $(i = 1, 2, \dots, n)$ be the following "rectangular waves" (see Figure 8):

$$\varphi_i(u) = \begin{cases} 1, & \theta_i \le u \le \theta_{i+1}, \\ 0, & \text{otherwise}, \end{cases} \qquad i = 1, 2, \dots, n, \tag{16}$$

then $F(x) = \sum_{i=1}^{n} \varphi_i(x) w_i$ is a piecewise zero-order interpolation function, where the graph of the interpolation composed of discrete steps (see Figure 9).

From Figure 9, we can see that, for a two-layer feedforward neural network and for given a group of data $\{(\theta_i, w_i)\}_{(1 \le i \le n)}$, the interpolation function $F(x) = \sum_{i=1}^{n} \varphi_i(x) w_i$ is based on the set of points (θ_i, w_i), $i = 1, 2, \dots, n$. Notice that, for such a network, the threshold values θ_i $(i = 1, 2, \dots, n)$ are the basic points of interpolation, the weights w_i $(i = 1, 2, \dots, n)$ are the values of the function to be interpolated, and the activation functions φ_i $(i = 1, 2, \dots, n)$ form the basic functions of interpolation. This means that the neurons of the hidden layer are just the basic functions of interpolation.

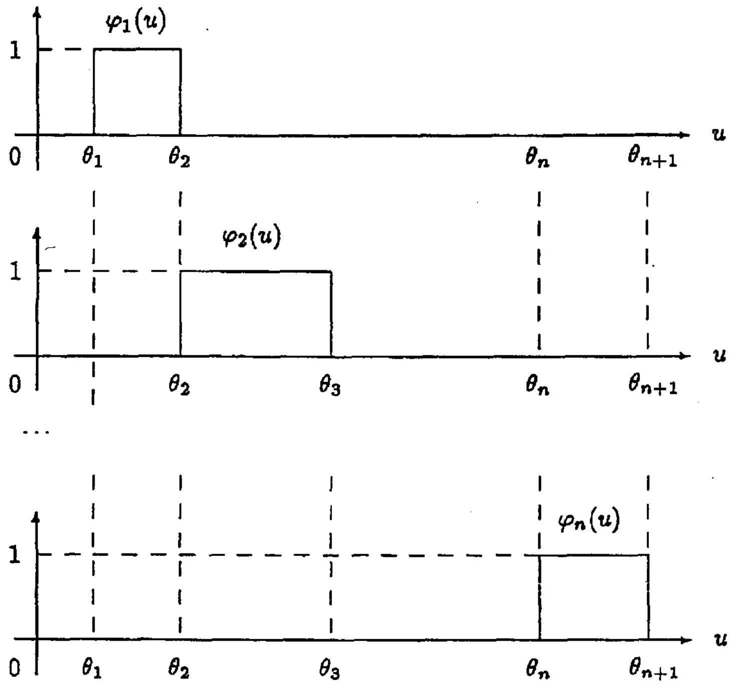

Figure 8. Rectangular waves as activation functions.

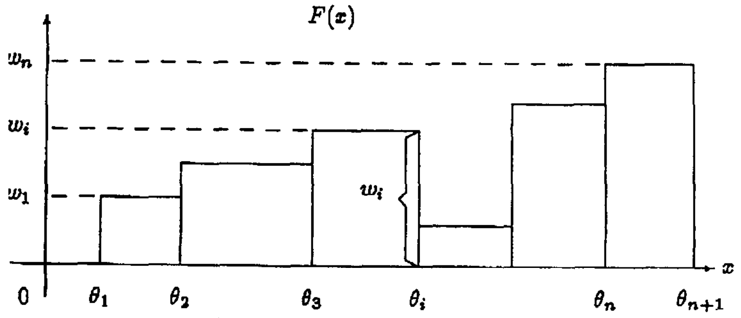

Figure 9. Piecewise zero-order interpolation function.

EXAMPLE 2. In equation (15), let the activation functions φ_i $(i = 1, 2, \dots, n)$ be the following "triangle waves" (see Figure 10):

$$\varphi_1(u) = \begin{cases} \dfrac{\theta_2 - u}{\theta_2 - \theta_1}, & \theta_1 \le u \le \theta_2, \\ 0, & \text{otherwise}, \end{cases} \tag{17}$$

$$\varphi_i(u) = \begin{cases} \dfrac{u - \theta_{i-1}}{\theta_i - \theta_{i-1}}, & \theta_{i-1} \le u \le \theta_i, \\ \dfrac{\theta_{i+1} - u}{\theta_{i+1} - \theta_i}, & \theta_i < u \le \theta_{i+1}, \\ 0, & \text{otherwise}, \end{cases} \tag{18}$$

where $i = 2, 3, \dots, n-1$.

$$\varphi_n(u) = \begin{cases} \dfrac{u - \theta_{n-1}}{\theta_n - \theta_{n-1}}, & \theta_{n-1} \le u \le \theta_n, \\ 0, & \text{otherwise}, \end{cases} \tag{19}$$

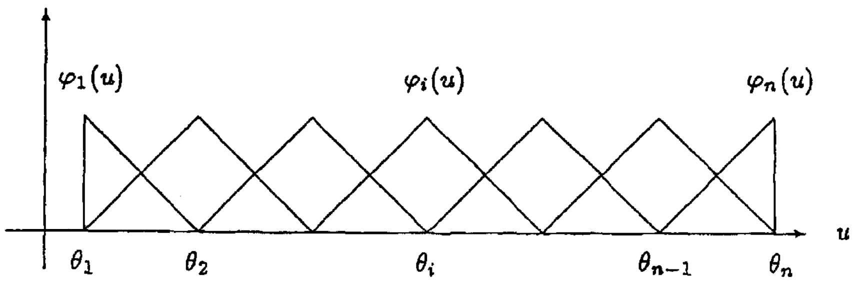

Figure 10. Triangle waves as activation functions.

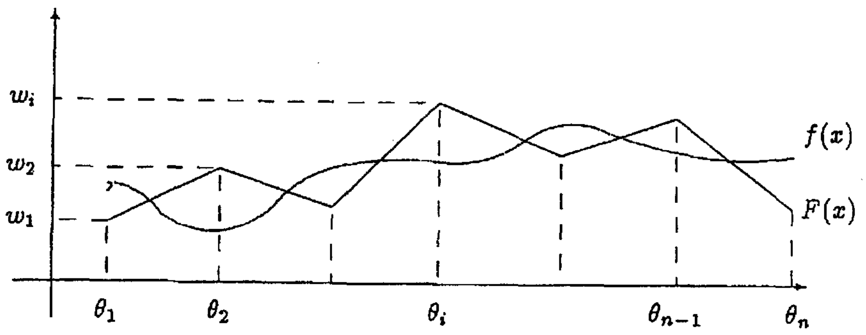

Figure 11. Piecewise linear interpolation function.

then $F(x) = \sum_{i=1}^{n} \varphi_i(x)w_i$ is a piecewise linear interpolation function, where the graph of the interpolation function composed of polygonal lines (see Figure 11).

The most important aspect of neural network is its learning ability, which is achieved by adjusting the connecting weights. From the standpoint of interpolation (see Figure 11), the adjustment of weights is the adjustment of the values of w_i so that the representation can approximate $f(\theta_i)$, $(i = 1, 2, \dots, n)$, sufficiently such that the interpolation function $F(x)$ can sufficiently represent the desired function $f(x)$. Furthermore, the threshold values θ_i, $i = 1, 2, \dots, n$ can also join the learning process. The values of θ_i, $i = 1, 2, \dots, n$, can be adjusted so that optimal positions can be obtained such that $F(x)$ can approximate $f(x)$ more accurately.

From the above discussions, we can obtain the following conclusion: *given any continuous function $y = f(x)$, there exists a two-layer neural network as that shown in Figure 7 such that the network can approximate this function $f(x)$ to any given degree of precision.*

4. THREE-LAYER FEEDFORWARD NEURAL NETWORK WITH TWO-INPUT ONE-OUTPUT

Figure 12 illustrates a three-layer feedforward neural network with two-input one-output, where the neurons h_{ij} are assumed to be hyperbolic functions, i.e., $h_{ij}(u,v) = uv$ $(i = 1, \dots, n,\ j = 1, \dots, m)$, and the neuron h is assumed to be the summation Σ, that is,

$$h(u_{11}, u_{12}, \dots, u_{nm}) = \Sigma(u_{11}, u_{12}, \dots, u_{nm}) = \sum_{i=1}^{n} \sum_{j=1}^{m} w_{ij} u_{ij}.$$

Let all the connection weights from f_i and g_j to h_{ij} $(i = 1, \dots, n,\ j = 1, \dots, m)$ be 1, then the

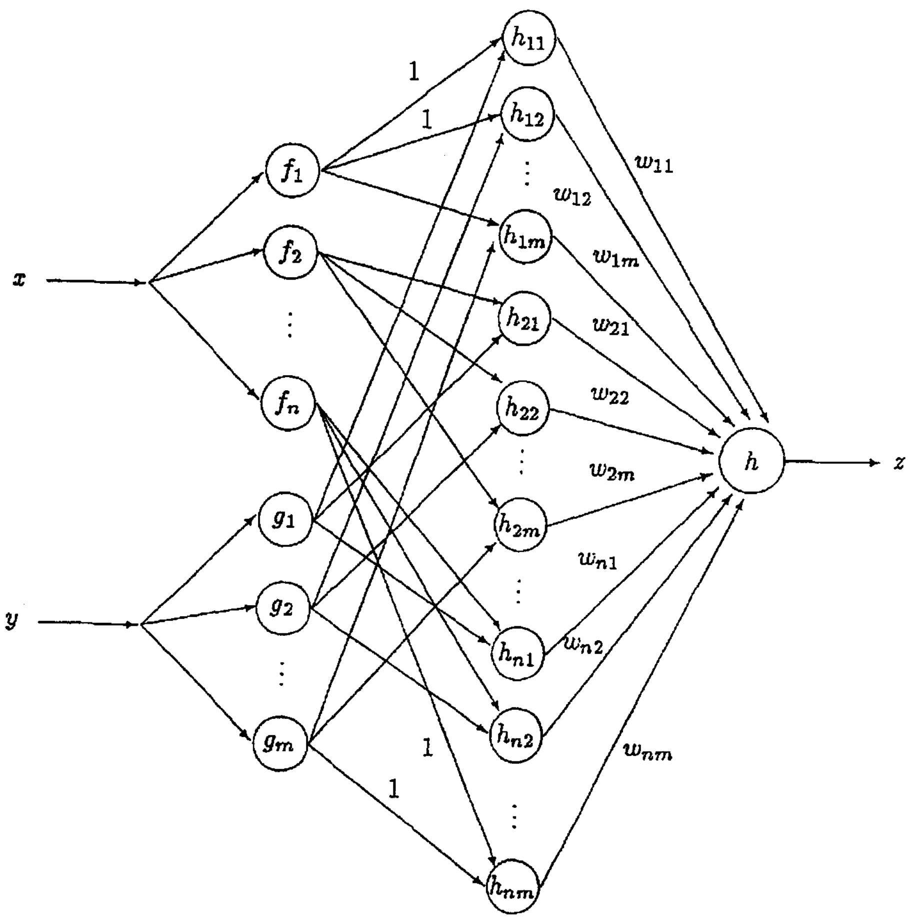

Figure 12. Three-layer feedforward neural network.

output of the network can be represented as follows:

$$\begin{aligned} F(x,y) = z &= h\left(h_{11}\left(f_1(x), g_1(y)\right), \dots, h_{1m}\left(f_1(x), g_m(y)\right), \dots,\right. \\ &\quad \left. h_{n1}\left(f_n(x), g_1(y)\right), \dots, h_{nm}\left(f_n(x), g_m(y)\right)\right) \\ &= h\left(f_1(x) g_1(y), \dots, f_1(x) g_m(y), \dots, f_n(x) g_1(y), \dots, f_n(x) g_m(y)\right) \\ &= \sum_{i=1}^{n} \sum_{j=1}^{m} w_{ij} f_i(x) g_j(y), \end{aligned} \tag{20}$$

which is a typical piecewise bivariate interpolation function formula.

EXAMPLE 3. Let the function $f_i(x)$ in equation (20) defined by equation (16), and the function $g_j(y)$ defined as

$$g_j(y) = \begin{cases} 1, & \delta_j \le y \le \delta_{j+1}, \\ 0, & \text{otherwise}, \end{cases} \tag{21}$$

where $j = 1, 2, \dots, m$, δ_j and δ_{j+1} (cf. Section 2) are the thresholds of the neuron g_j ($j = 1, 2, \dots, m$), then the function $F(x, y)$ determined by equation (20) is a piecewise bivariate zero-order interpolation function, which can approximate the function $f(x, y)$ by adjusting the weights w_{ij}.

EXAMPLE 4. In equation (20), let the functions $f_i(x)$ and $g_j(y)$ be triangle wave activation functions. In other words, the functions $f_i(x)$ are defined by equations (17)–(19), and the functions $g_j(y)$ are also defined by equations (17)–(19) except that θ_i is replaced by δ_j, and n by m. Now, $F(x, y)$ becomes a piecewise bivariate linear interpolation function.

Following Section 3, we can obtain a second conclusion: *given any bivariate continuous function $z = f(x, y)$, there exists a three-layer neural network as that shown in Figure 12 such that the network can approximate this function $f(x, y)$ to any degree of precision.*

5. LEARNING ALGORITHMS OF THE FEEDFORWARD NEURAL NETWORKS

Begin with a simple case, consider the learning algorithm of the two-layer feedforward neural network with one input and one output as that shown in Figure 7. The relation between the input and output of the network is

$$F(x) = \sum_{i=1}^{n} \varphi_i(x)\, w_i. \tag{22}$$

Consider supervised learning with a group of training sample as

$$\{(x_i, y_i) \mid i = 1, 2, \dots, p\}. \tag{23}$$

Substituting the training samples into equation (22), we obtain a system of linear equations with w_j $(j = 1, 2, \dots, m)$ as the unknowns:

$$\begin{aligned} \varphi_1(x_1)\, w_1 + \varphi_2(x_1)\, w_2 + \cdots + \varphi_n(x_1)\, w_n &= y_1, \\ \varphi_1(x_2)\, w_1 + \varphi_2(x_2)\, w_2 + \cdots + \varphi_n(x_2)\, w_n &= y_2, \\ &\vdots \\ \varphi_1(x_p)\, w_1 + \varphi_2(x_p)\, w_2 + \cdots + \varphi_n(x_p)\, w_n &= y_p. \end{aligned} \tag{24}$$

Let $a_{ij} = \varphi_j(x_i)$, $A = (a_{ij})_{p\times n}$, $Y = (y_1, y_2, \dots, y_p)^\top$, and $W = (w_1, w_2, \dots, w_n)^\top$, then equation (24) can be simplified as

$$AW = Y. \tag{25}$$

Since W is unknown in equation (25), we usually try to find a $W^* = (w_1^*, w_2^*, \dots, w_n^*)^\top$ such that AW approximates Y sufficiently close according to some sort of norm $\|\cdot\|$, such as

$$\min \|AW - Y\|. \tag{26}$$

Therefore, it becomes a problem of optimization.

In the field of neural network, the norm $\|\cdot\|$ is often taken as the following form:

$$E(w_1, \dots, w_n) = \frac{1}{2}\sum_{i=1}^{p} (\hat{y}_i - y_i)^2 = \frac{1}{2}\sum_{i=1}^{p} \left(\sum_{j=1}^{n} a_{ij} w_j - y_j\right)^2. \tag{27}$$

This is an optimization problem with quadratic objective function and without any constraint, which can usually be solved by the gradient descent algorithm. To solve equation (26), let the gradient be

$$\nabla E(W) = \left(\frac{\partial E(W)}{\partial w_1}, \frac{\partial E(W)}{\partial w_2}, \dots, \frac{\partial E(W)}{\partial w_n}\right)^\top. \tag{28}$$

Assume that we have carried out k iterations and obtained the k^{th} result W_k. Then the $(k+1)^{\text{th}}$ iteration result W_{k+1} can be obtained by the use of a linear search along the the gradient direction of $-\nabla E(W_k)$, i.e.,

$$W_{k+1} = W_k - \mu_k \nabla E(W_k), \tag{29}$$

where the step size factor μ_k satisfies the following optimal condition:

$$E\left(W_k - \mu_k \nabla E\left(W_k\right)\right) = \min_{\mu} E\left(W_k - \mu \nabla E\left(W_k\right)\right). \tag{30}$$

Combine equations (29) and (30), we have

$$W_{k+1} = \mathrm{ls}\left(W_k, -\nabla E\left(W_k\right)\right), \tag{31}$$

where "l" and "s" represent the first letter of "linear" and "search", respectively.

Thus, we obtain a sequence of points $W_0, W_1, W_2, \ldots$, where W_0 is the initial point that can be chosen arbitrarily. Hopefully, this sequence of points $\{W_k\}$ converges to the minimum point W^*.

REMARK 2. To reduce computation, the linear search is usually not carried out in the literature. The learning algorithm is frequently some sort of simplified form of equation (29). For example, denoting $\frac{\partial E}{\partial W} = \nabla E(W)$, then we have the following equation:

$$W_{k+1} = W_k + \mu \left(-\frac{\partial E}{\partial W}\right)_{W=W_k}, \tag{32}$$

which is just one of the algorithms in common use. It is a simplified form of equation (29), because μ is chosen in advance instead of obtaining μ_k by minimization along the gradient.

REMARK 3. It is easy to show that $(-\nabla E(W_{k+1}))(-\nabla E(W_k)) = 0$ which means that $-\nabla E(W_{k+1})$ is perpendicular to $-\nabla E(W_k)$. Thus, the gradient directions form a "sawtooth". This phenomenon is well known in the gradient optimization literature. This is one of the principle reasons why neural networks generally converge slowly.

REMARK 4. The quadratic equation, equation (27) can be rearranged as follows:

$$\begin{aligned}
E(W) &= \frac{1}{2}\sum_{i=1}^{p}\left(\sum_{j=1}^{n} a_{ij}w_j - y_i\right)^2 \\
&= \frac{1}{2}\sum_{i=1}^{p}\left[\left(\sum_{j=1}^{n} a_{ij}w_j\right)^2 - 2y_i\sum_{j=1}^{n} a_{ij}w_j + y_i^2\right] \\
&= \frac{1}{2}\left[\left(\sum_{i=1}^{p} a_{i1}^2\right)w_1^2 + \left(\sum_{i=1}^{p} a_{i2}^2\right)w_2^2 + \ldots + \left(\sum_{i=1}^{p} a_{in}^2\right)w_n^2 + 2\left(\sum_{i=1}^{p} a_{i1}a_{i2}\right)w_1w_2\right. \\
&\quad + 2\left(\sum_{i=1}^{p} a_{i1}a_{i3}\right)w_1w_3 + \ldots + 2\left(\sum_{i=1}^{p} a_{i1}a_{in}\right)w_1w_n + 2\left(\sum_{i=1}^{p} a_{i2}a_{i3}\right)w_2w_3 \\
&\quad \left. + 2\left(\sum_{i=1}^{p} a_{i2}a_{i4}\right)w_2w_4 + \ldots + 2\left(\sum_{i=1}^{p} a_{i2}a_{in}\right)w_2w_n + \ldots + 2\left(\sum_{i=1}^{p} a_{i,n-1}a_{in}\right)w_{n-1}w_n\right] \\
&\quad + \left(-\sum_{i=1}^{p} a_{i1}y_i\right)w_1 + \left(-\sum_{i=1}^{p} a_{i2}y_i\right)w_2 + \ldots + \left(-\sum_{i=1}^{p} a_{in}y_i\right)w_n + \frac{1}{2}\sum_{i=1}^{p} y_i^2 \\
&= \frac{1}{2}W^{\mathrm{T}}QW + B^{\mathrm{T}}W + C,
\end{aligned} \tag{33}$$

where

$$\begin{aligned}
Q &= \left(q_{ij}\right)_{n\times n}, \\
B &= \left(b_1, b_2, \ldots, b_n\right)^{\mathrm{T}}, \\
C &= \frac{1}{2}\sum_{i=1}^{p} y_i^2, \\
q_{ij} &= \sum_{k=1}^{p} a_{ki}a_{kj},
\end{aligned}$$

with $q_{ij} = q_{ji}$ and $b_i = -\sum_{k=1}^{p} a_{ki} y_k$. Clearly, we have

$$\nabla E(W) = QW + B. \tag{34}$$

To obtain the stationary point of $E(W)$, let $\nabla E(W) = 0 \triangleq (0, 0, \dots, 0)^{\mathrm{T}}$. When Q is a positive definite matrix, the unique stationary point can be obtained as follows:

$$W^* = -Q^{-1}B. \tag{35}$$

The Hesse matrix at this point is

$$\nabla^2 E(W^*) = \nabla(\nabla E(W^*)) = \begin{bmatrix} \dfrac{\partial^2 E(W)}{\partial w_1^2} & \dfrac{\partial^2 E(W)}{\partial w_2 \partial w_1} & \cdots & \dfrac{\partial^2 E(W)}{\partial w_n \partial w_1} \\ \dfrac{\partial^2 E(W)}{\partial w_1 \partial w_2} & \dfrac{\partial^2 E(W)}{\partial w_2^2} & \cdots & \dfrac{\partial^2 E(W)}{\partial w_n \partial w_2} \\ \cdots & \cdots & \cdots & \cdots \\ \dfrac{\partial^2 E(W)}{\partial w_1 \partial w_n} & \dfrac{\partial^2 E(W)}{\partial w_2 \partial w_n} & \cdots & \dfrac{\partial^2 E(W)}{\partial w_n^2} \end{bmatrix}_{w=w^*} = Q,$$

which is a positive definite matrix because Q is positive definite. From optimization theory, we know that $W^* = -Q^{-1}B$ is the unique minimal point of the problem.

REMARK 5. Since the learning algorithm, equation (31) or equations (29) and (30), is in an implicit form, it is not very convenient to use. Let us consider the explicit form. From Remark 3, we known that $(\nabla E(W_{k+1}))^{\mathrm{T}} \nabla E(W_k) = 0$. Substituting equation (34) and then equation (29) into this expression, we have

$$(Q(W_k - \mu_k \nabla E(W_k)) + B)^{\mathrm{T}} (QW_k + B) = 0.$$

Rearranging this equation, we have

$$(\nabla E(W_k) - \mu_k Q \nabla E(W_k))^{\mathrm{T}} \nabla E(W_k) = 0.$$

Solving for μ_k, we have

$$\mu_k = \frac{(\nabla E(W_k))^{\mathrm{T}} \nabla E(W_k)}{(\nabla E(W_k))^{\mathrm{T}} Q \nabla E(W_k)}, \tag{36}$$

substituting into equation (29), we finally have the explicit scheme:

$$W_{k+1} = W_k - \frac{(\nabla E(W_k))^{\mathrm{T}} \nabla E(W_k)}{(\nabla E(W_k))^{\mathrm{T}} Q \nabla E(W_k)} \nabla E(W_k). \tag{37}$$

Now, we turn to consider the learning algorithm of the forward neural network with two-input one-output. From Section 4, the relation between the input and output of the network is as follows:

$$F(x, y) = \sum_{i=1}^{n} \sum_{j=1}^{m} f_i(x) g_j(y) w_{ij}. \tag{38}$$

Substituting the following into the above equation, given a group of training samples:

$$\{((x_k, y_k), z_k) \mid k = 1, 2, \dots, p\}, \tag{39}$$

we obtain the following system of linear equations with w_{ij} $(i = 1, 2, \dots, n,\ j = 1, 2, \dots, m)$ as unknowns:

$$\sum_{i=1}^{n} \sum_{j=1}^{m} f_i(x_k) g_j(y_k) w_{ij} = z_k, \qquad k = 1, 2, \dots, p. \tag{40}$$

Let $a_{kij} = f_i(x_k)g_j(y_k)$. Then above equation can be rewritten as

$$\sum_{i=1}^{n}\sum_{j=1}^{m} a_{kij}w_{ij} = z_k, \qquad k = 1,2,\dots,p. \tag{41}$$

To simplify the triple subscripts of above equation, let

$$l = (i-1)\,m + j. \tag{42}$$

Then equation (41) takes the following form:

$$\sum_{l=1}^{q} a_{kl}w_l = z_k, \qquad k = 1,2,\dots,p, \tag{43}$$

where $q = nm$. Furthermore, if we let $A = (a_{kl})_{p\times q}$, $Z = (z_1, z_2, \dots, z_p)^\top$, and $W = (w_1, w_2, \dots, w_q)^\top$, then equation (43) becomes the following matrix equation:

$$AW = X, \tag{44}$$

which is similar to equation (25). Thus, the learning algorithm of equation (44) should be similar to that of equation (25).

6. FEEDFORWARD NEURAL NETWORK WITH MULTI-INPUT MULTI-OUTPUT AND ITS LEARNING ALGORITHM

The three-layer feedforward neural network with two inputs and one output (see Figure 12) can be extended easily to a network with two inputs and two outputs. Let the two output nodes be represented by $h01$ and $h02$ with the weight vectors between the hiding layer and these output nodes as w^1 and w^2, respectively, the relation between the input and the output of this network can be obtained in a similar fashion as that shown in Section 4. This input-output relationship is

$$F(x,y) = (u,v) = \left(\sum_{i=1}^{n}\sum_{j=1}^{m} f_i(x)\,g_j(y)\,w_{ij}^{(1)}, \sum_{i=1}^{n}\sum_{j=1}^{m} f_i(x)\,g_j(y)\,w_{ij}^{(2)}\right), \tag{45}$$

where u and v are the two outputs from the two output nodes. The above input-output relation is a vector-valued interpolation function.

Following Section 4, we can obtain a third conclusion: *given any multivariate vector-value continuous function $f(x_1, x_2, \dots, x_n) = (y_1, y_2, \dots, y_m)$, there exists a three-layer feedforward neural network such that the network can approximate this function $f(x_1, x_2, \dots, x_n)$ to any given degree of precision.*

To consider the learning algorithm of this network, let us assume that we have the following given group of samples:

$$\{((x_k, y_k), (u_k, v_k)) \mid k = 1,2,\dots,p\}. \tag{46}$$

Substituting this group of samples into equation (45), we obtain the following two systems of linear equations:

$$\sum_{i=1}^{n}\sum_{j=1}^{m} f_i(x_k)\,g_j(y_k)\,w_{ij}^{(1)} = u_k, \qquad k = 1,2,\dots,p, \tag{47}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{m} f_i(x_k)\,g_j(y_k)\,w_{ij}^{(2)} = u_k, \qquad k = 1,2,\dots,p. \tag{48}$$

Let $a_{kij} = f_i(x_k)g_j(y_k)$, $l = (i-1)m + j$, and $q = nm$, then above two equations become

$$\sum_{l=1}^{q} a_{kl} w_l^{(1)} = u_k, \qquad k = 1, 2, \dots, p, \tag{49}$$

$$\sum_{l=1}^{q} a_{kl} w_l^{(2)} = v_k, \qquad k = 1, 2, \dots, p. \tag{50}$$

Let

$$\begin{aligned} A &= (a_{kl})_{p\times q}, \\ W_1 &= \left(w_1^{(1)}, w_2^{(1)}, \dots, w_q^{(1)}\right)^{\top}, \\ W_2 &= \left(w_1^{(2)}, w_2^{(2)}, \dots, w_q^{(2)}\right)^{\top}, \\ U &= (u_1, u_2, \dots, u_p)^{\top}, \\ V &= (v_1, v_2, \dots, v_p)^{\top}, \end{aligned}$$

then equations (49) and (50) can be written in the following matrix forms:

$$AW_1 = U, \qquad AW_2 = V, \tag{51}$$

which is similar to equation (25).

7. CONCLUSION

We have discussed the feedforward neural networks from the mathematical point of view. First, the interpolative representations and structures of the various feedforward neural networks are given and discussed in detail. Then some approximation properties are summarized. Finally, the learning algorithms of the feedforward neural networks are analyzed. We also obtained the explicit form from the usually implicit learning algorithm of the neural network.

REFERENCES

1. J. Hertz, A. Krogh and R.G. Palmer, *Introduction to the Theory of Neural Computation*, Addison-Wesley, New York, (1991).
2. B. Kosko, *Neural Networks and Fuzzy Systems*, Prentice-Hall, Englewood Cliffs, NJ, (1992).
3. H.X. Li, Multifactorial functions in fuzzy sets theory, *Fuzzy Sets and Systems* **35**, 69–84, (1990).
4. H.X. Li and V.C. Yen, *Fuzzy Sets and Fuzzy Decision-Making*, CRC Press, Boca Raton, FL, (1995).
5. H.X. Li, To see the success of fuzzy logic from mathematical essence of fuzzy control—On "The paradoxical success of fuzzy logic", *Fuzzy Systems and Mathematics* **9** (4), 1–14, (1995).
6. H.X. Li, Multifactorial fuzzy sets and multifactorial degree of nearness, *Fuzzy Sets and Systems* **19**, 291–297, (1986).
7. H.X. Li, Interpolation mechanism of fuzzy control, *Science in China (Series E)* **41** (3), 312–320, (1998).
8. H.X. Li, *Fuzzy Information Processing and Fuzzy Computers*, Science Press, New York, (1997).
9. H.X. Li and C.L.P. Chen The equivalence between fuzzy logic systems and feedforward neural networks, *IEEE Transactions on Neural Networks* **11** (2), 356–365, (2000).
10. Y.H. Pao, *Adaptive Pattern Recognition and Neural Networks*, Addison-Wesley, New York, (1989).