NEURAL NETWORKS FOR PREDICTION
AND CONTROL OF DISCRETE SYSTEMS

Y. Jin, A.G.Pipe, A.Winfield

Faculty of Engineering, University of the West of England
Frenchay Campus, Coldharbour Lane, Bristol BS16 1QY, UK

Abstract

This paper describes how to use neural networks in prediction and control of discrete nonlinear systems. First the concept of Linear Equivalent (L-E) neural networks is introduced. Based on the general form of L-E neural networks, prediction and control methods are developed. Neural network learning algorithms are carefully designed so that it is theoretically guaranteed that the predicted value will follow the actual system output for prediction systems and that in control the actual system output will follow the desired value. Discrete nonlinear systems fall into two classes, i.e.

$$y_{k+1} = \sum_{i=0}^{n} a_i y_{k-i} + \sum_{i=0}^{m} b_i \mu_{k-i} + f(y_k, \ldots, y_{k-i})$$

and

$$y_{k+1} = b(y_k, \ldots, y_{k-i}) u_k + f(y_k, \ldots, y_{k-i})$$

Examples are employed to explain the theoretical results. Further extensions of the results and the difficulties involved are finally discussed.

Keywords - Neural Networks, Prediction, Control, Discrete Systems, Nonlinear Systems

1. INTRODUCTION

It is well known that neural networks have three features, i.e. universal approximation, learning ability and parallel computing architectures. These features are very attractive for solving complex engineering problems, e.g. nonlinear system prediction and control. By universal approximation, neural networks are able to predict and control nearly all nonlinear systems. Through learning neural networks can improve their performance and
finally achieve satisfactory prediction and control results. Parallel computational architectures can help ease the real time computing problem of highly complicated control algorithms.

Much research effort has been put into the design of neural network applications for nonlinear system prediction and control. However most of results are only at an empirical level. The main reason is that whilst most simulation results are very good there has been a scarcity of proven theories for guaranteed safety, although this situation has improved over the last three years. In Sanner & Slotine, 1992, and Hancock & Fallside, 1991, the authors theoretically established a stable result for neural adaptive control of a certain class of continuous nonlinear system. At nearly the same time and using the similar technology, Jin et al., 1992 & 1993, independently established stable neural network control and prediction results for discrete nonlinear systems. Since neural networks are also discrete, they might be realized by using digital VLSI technology resulting in a cheap and realisable implementation. This paper presents our recent results in this effort. The nonlinear systems discussed here include the following two classes.

\[
\begin{align*}
\text{Class 1:} & \quad y_{k+1} = \sum_{i=0}^{a} a_i y_{k-i} + \sum_{i=0}^{m} b_i \mu_{k-d-i} + f(y_k, \ldots, y_{k-1}) \\
\text{Class 2:} & \quad y_{k+1} = b(y_k, \ldots, y_{k-1}) u_k + f(y_k, \ldots, y_{k-1})
\end{align*}
\] (1) (2)

In the following section, we first introduce the concept of Linear-Equivalent (L-E) neural networks. The neural network prediction and control results are based on the general form of L-E neural networks, therefore the results are suitable for any type of neural network in the L-E group. Section 3 presents prediction of nonlinear systems. The control results are presented in Section 4. The last section presents conclusions and discussions. Further extensions and their difficulties are also included in this last section. During the theoretical development, simulation examples are also presented to explain the theoretical results. Proofs of all theorems are presented in the appendices.

2. LINEAR-EQUIVALENT NEURAL NETWORKS

There now exist many different neural network structures since the detailed structures depend on their applications. In this section, from the viewpoint of control applications, we classify neural networks into two groups, i.e. Linear Equivalent (L-E) neural networks and Back-Propagation Equivalent (BP-E) neural networks or Nonlinear-Equivalent (NL-E) neural networks. The criteria for this classification is that in L-E neural networks output is linear with adjustable weights while in BP-E neural networks output is nonlinear with adjustable weights. Neural networks are classified in this way simply because there is a large difference between linear and nonlinear systems from the
perspective of progress in control theory. L-E neural networks correspond to linear systems and BP-E neural networks correspond to nonlinear systems. In this section we present three main types of L-E neural networks, i.e. Radial Basis Function (RBF) neural networks, Cerebellar Model Articulation Controller (CMAC) neural networks, Adaptive Fuzzy Controllers (AFC). They can all be viewed as three layer neural networks (see Figure 1) where the hidden layer performs a fixed nonlinear mapping. The best advantage of L-E neural networks is that we can guarantee stable results.

2.1 Radial Basis Function (RBF) Neural Networks

It is well established that any continuous function can be expressed as an expansion of a series of basis functions, i.e. $f(x) = \sum_{i=0}^{M} w_i g_i(x)$ where $x$ is the input space and $g_i$ is a basis function. This idea has recently been used to build neural networks. The hidden layer performs a fixed nonlinear transformation which maps the input space into the radial basis function space $g(x) = (g_0, \ldots, g_M)^T$. The radial basis function vector $g$ is determined by the neural network structure. The output layer then linearly combines these basis functions. The vector of coefficients in the combination forms the neural network weights.

Girosi & Poggio, 1990, and Sanner & Slotine, 1992, fully explained radial basis function neural networks and also described how to choose radial basis functions. A common set of basis functions are the Gaussian radial basis functions, i.e. $e^{-\sum_{l=1}^{n} (x_i - x_{i,l})^2}$ where $x_1, \ldots, x_n$ are the $n$ neural network inputs and $(x_{i,1}, \ldots, x_{i,m})$ is the basis function...
centres. The basis function vector, \( \mathbf{g} \), consists of all basis functions (nodes) whose centres locate on a grid with distance \( \Delta \) in each input axis. Normally \( \Delta \) and \( \sigma_v \) are chosen having \( \Delta \sigma_v = \frac{\pi}{2\theta} \) where \( \theta \) is called the "oversampling parameter" in Sanner & Slotine, 1992.

Now we analyze the norm of the basis function vector \( \mathbf{g} \). When the number of inputs is 1, \( \|\mathbf{g}\|^2 \leq \sum_{i=0}^{M} g_i \) since elements of \( \mathbf{g} \) are positive and less than or equal to 1. For \( \sum_i g_i \) we have

\[
\sum_i g_i \leq 2 + 2e^{-\Delta \alpha} + 2e^{-2\Delta \alpha} + 2e^{-3\Delta \alpha} + \ldots \\
\leq 2 + 2 \frac{e^{-\Delta \alpha}}{1 - e^{-3\Delta \alpha}}
\]

When neural networks have multiple inputs we also have \( \|\mathbf{g}\|^2 \leq \sum_i g_i \) and

\[
\sum_i g_i = \sum_i e^{-\Delta \alpha \sum_{j=1}^{n} x_{ij}} \\
= \prod_{i=1}^{n} \sum_i e^{-\Delta \alpha x_{ij}} \quad \text{(4)}
\]

From the analysis of the single input neural networks we have

\[
\sum_{i=0}^{M} g_i \leq \left[ 2 + 2 \frac{e^{-\Delta \alpha}}{1 - e^{-3\Delta \alpha}} \right]^{n}
\]

Therefore we are able to conclude that for either single input or multiple input neural networks the basis function vector has a bounded norm.

There could be several variations in designing RBF neural network structures. For example, it is not necessary that the centres of Gaussian nodes are spread evenly in the input axes. Other examples are that the variances \( \sigma_v \) for each axis could be different and that even for one axis the variances could be differ throughout the region.
2.2 Cerebellar Model Articulation Controller (CMAC) Neural Networks

CMAC models were first introduced by Albus, 1975. He successfully applied them in robot control. Miller et al., 1990, and Kraft et al., 1992, extended CMAC in robot control applications. They presented learning algorithms and compared the CMAC control results with other conventional adaptive control results.

The basic diagram of CMAC is shown in Figure 2. The CMAC neural network learns a function \( f(x) \) by associating measurements of \( x \) with \( f(x) \). The first layers of the network determine a mapping of the input vector \( x \) to particular weights in the network. Each input vector is associated with a particular set of weights. The size of the set is normally fixed.

The input sensors detect the inputs \( x \). According to the inputs, the state space detectors decide which states are active. The size of the state space will become huge when the number of inputs is large so a random (but fixed) mapping similar to hash-coding is used, which maps from the state space to a multiple field or commonly the physical memory space. All these are fixed for a particular neural network, i.e. the mapping from \( x \) to \( g(x) \) is fixed during learning. \( g \) is a binary vector. Although \( g \) changes with inputs the total number of 1s stays the same at all times therefore \( g \) has a bounded norm. The data stored in the memory are the neural network weight values. The data corresponding to active states are the weights associated with the input vector. The sum of these weight values is the neural network output.

2.3 Adaptive Fuzzy Controllers

A fuzzy logic system consists of three parts: fuzzifier, defuzzifier, and fuzzy inference engine working on fuzzy rules. It is shown in Figure 3. A typical fuzzy rule is
$R^\omega: \text{ IF } x_1 \text{ is } F_1 \text{ and } \ldots \text{ and } x_n \text{ is } F_n \text{ THEN } y \text{ is } G$

There are several implementational choices for each part of a fuzzy logic system. Fuzzy logic systems with centre-average defuzzifier, product inference, and singleton fuzzifier are of the following form:

$$y(x) = \frac{\sum_{i=1}^{M} \bar{y}'(\Pi_{i=1}^{n} \mu_{y_i}(x_i))}{\sum_{i=1}^{M} (\Pi_{i=1}^{n} \mu_{y_i}(x_i))}$$  \hspace{1cm} (6)

where $\bar{y}'$ is the point at which $\mu_{G_i}$ has its maximum value which is assumed to be 1.

Basic fuzzy logic systems are fixed, non-adaptive. However fuzzy controllers are often used to work in situations where there is a large uncertainty or unknown variation in plant parameters and structures. Therefore advanced fuzzy control should be adaptive in order to maintain consistent performance of a system where uncertainties are present.

![Figure 3: Fuzzy Logic Systems](image)

One choice of adaptive parameters is $\bar{y}'$ in Equation 6 (Wang, 1993). In this case the adaptive fuzzy logic system is now a type of L-E neural network. The fixed nonlinear mapping in the hidden layer is $\frac{\Pi_{i=1}^{n} \mu_{y_i}(x_i)}{\sum_{i=1}^{M} (\Pi_{i=1}^{n} \mu_{y_i}(x_i))}$. The adjustable weights are $\bar{y}'$. It is clear that the nonlinear mapping vector of the hidden layer has a bounded norm.

To summarise we are able to say that a L-E neural network can be expressed in general form as

$$y = g(x)^T W$$  \hspace{1cm} (7)

where $g$ is the hidden layer nonlinear mapping vector which depends on the neural network structure as well as the neural network input $x$ and has a bounded norm, $W$ is
the weight vector.

There are other types of neural networks which can also be classified into this group, e.g. B-Spline neural networks (Lane et al., 1991). Nonlinear Equivalent neural networks can also be viewed as L-E neural networks if the neural network weights are near the desired weights. Let \( N(W) \) be the neural network function, \( W \) be the neural network vector and \( W^* \) be the desired neural network vector. Using the first order Taylor series expansion, we obtain

\[
N(W) = N(W^*) + \frac{\partial N(W^*)}{\partial W^*}(W - W^*)
\]  

(8)

Since \( W^* \) is fixed, the neural network output is linear with the neural network weights. This means that all results for L-E neural networks are suitable to NL-E neural networks if their initial weights are near the desired weights.

3. PREDICTION OF NONLINEAR DISCRETE SYSTEMS

In neural prediction the neural network and the plant are two separate systems. The neural network gets inputs from the plant and predicts what the plant outputs might be. Stability is not the issue in this case. However, it is necessary to theoretically guarantee that the neural network will accurately predict the plant outputs. We shall call the error between the predicted result and the actual system output as the prediction error.

3.1 Nonlinear Systems of Class 1

This subsection discusses the prediction for systems described by Equation 1 where \( a_i, b_i \) and the nonlinear function \( f \) are unknown. A neural network is used to approximate the nonlinear function \( f \), i.e.

\[
f(y_{k}, y_{k-1}, \ldots, y_{k-p}) = \sum_{i=0}^{M} g_i(y_{k}, \ldots, y_{k-i})w_i + e_{ak}
\]

(9)

where \( e_{ak} \) is the approximation error and satisfies \( |e_{ak}| \leq e_m \). \( e_m \) is called the approximation accuracy. Because \( f \) is unknown, the neural network weights, \( w_i \), are also unknown. Therefore we call \( w_i \) the desired neural network weights.

By substituting \( f \) with the neural network, Equation 1 is changed to

\[
y_{k+1} = \sum_{i=0}^{n} a_i y_{k-i} + \sum_{i=0}^{n} b_i \mu_{k-d-i} + \sum_{i=0}^{M} g_i w_i + e_{ak}
\]

(10)

From the above analysis we find that a neural network plus a linear system is capable of
predicting Equation 1. The prediction model is then

$$y_{k+1} = \sum_{i=0}^{n} a_i^k y_{k,i} + \sum_{i=0}^{m} b_i^k u_{k,d-i} + \sum_{i=0}^{M} g_i^k w_i$$  \hspace{1cm} (11)$$

where \(a_i^k, b_i^k\) are the kth estimations of \(a_i\) and \(b_i\), \(w_i^k\) are the kth estimations of the neural network desired weights, \(w_i\). The result is summarized in a theorem form as follows.

**THEOREM 1:** Provided that \(u_i\) and \(y_i\) are bounded the following combination of neural network learning algorithm and \(a_i, b_i\) estimation algorithms guarantees that the neural network weights (estimations) and estimations of \(a_i\) and \(b_i\) will approach fixed bounded values and the final prediction error will not be greater than \(e_m\) which is defined in Equation 12.

Define

$$e_m = \frac{2e_m}{\beta - \alpha} \hspace{1cm} (12)$$

$$c_k = 1 + \sum_{i=0}^{n} y_{k,i}^2 + \sum_{i=0}^{m} u_{k,d-i}^2 + \sum_{i=0}^{M} g_i^2 \hspace{1cm} (13)$$

$$y_{k+1} = y_{k+1} - y_{k+1} \hspace{1cm} (14)$$

$$e_m = \begin{cases} 
0 & \text{if } |y_{k+1}| > e_m \\
|y_{k+1}| & \text{if } |y_{k+1}| \leq e_m 
\end{cases} \hspace{1cm} (15)$$

The neural network learning algorithm

$$w_i^{k+1} = w_i^k + \frac{\alpha}{c_k} g_i e_m \hspace{1cm} i=0,1,\ldots,M \hspace{1cm} (16)$$

The \(a_i\) estimation algorithm

$$a_i^{k+1} = a_i^k + \frac{\alpha}{c_k} y_{k,i} e_m \hspace{1cm} i=0,1,\ldots,n \hspace{1cm} (17)$$

The \(b_i\) estimation algorithm

$$b_i^{k+1} = b_i^k + \frac{\alpha}{c_k} u_{k,d-i} e_m \hspace{1cm} i=0,1,\ldots,m \hspace{1cm} (18)$$

Where \(\alpha, \beta\) are constants and \(0 < \alpha < \beta < 2\). \(\alpha\) is also called the learning rate.

**Remark 1:** The final prediction error \(e_m\) is related to the neural network approximation accuracy \(e_m\). The neural network approximation accuracy could be as high as required by carefully constructing the neural network. Therefore the prediction error could also
be reduced to any required accuracy.

**Remark 2:** The theorem only guarantees that the prediction error will decrease to a certain level. It does not say anything about the error between the neural network output and the nonlinear function. It also does not say whether the neural network learns the desired weights. All these factors depend on the quality of the vector $g$, i.e. whether $g$ is persistent exciting. This remark is also valid in the following sections.

**Example 1:** The system discussed here is

$$y_{k+1} = 0.8y_k - 0.16y_{k-1} + u_k + 0.8u_{k-1} + f(y_k, y_{k-1}) \quad \text{where} \quad f = \frac{5y_k y_{k-1}}{1 + y_k^2 + y_{k-1}^2}.$$ 

The control input function is $u_k = \sin(2\pi k/25)$. Comparing the system with Equation 1, we have $a_0 = 0.8$, $a_1 = -0.16$, $b_0 = 1$, $b_1 = 0.8$, and $d = 0$. We also know that the system output is in the range $(1.7, 11.5)$.

First we build a neural network for approximation of the nonlinear function $f$. A RBF neural network with Gaussian basis functions is used. The neural network has two inputs $y_k$ and $y_{k-1}$. The centres of its Gaussian nodes locate on a grid with distance 0.1 in each input axis. $\sigma^2 = 39.27$. Only the nodes whose centres are in the range $[1.2, 12] \times [1.2, 12]$ are used. Therefore the neural network has 11,881 nodes in total. The neural network approximation accuracy is 0.001.

Then we choose that $\beta = 1.9$, $\alpha = 0.5$. 2000 points in the time domain are simulated. The first 100 points are shown in Figures 4a. The error is quite small. However the estimated parameters are far from the actual values, which in the 100th running point are $a_0 = 0.73815$, $a_1 = 0.31797$, $b_0 = -0.06979$, $b_1 = -0.21237$. The last 100 points are shown in Figure 4b. The prediction perfectly matches the

![Figure 4: System Output (solid line) and Its Prediction (dashed line). (a) The First 100 Points, (b) The Last 100 Points.](image-url)
actual system output. The estimated parameters are \( a_0 = 1.37137 \), \( a_1 = -0.406638 \), \( b_0 = 0.322069 \), \( b_1 = -0.125732 \).

**Remark 3:** There is a difficulty in applying the theorem result. Since the nonlinear function is unknown, the neural network approximation accuracy is also not known. Therefore it is difficult to calculate the deadzone. This difficulty is overcome by trial and error. This remark is also valid in the following examples.

### 3.2 Nonlinear Systems of Class 2

In this subsection we discuss the prediction of nonlinear systems described by Equation 2, where \( f \) and \( b \) are unknown nonlinear functions. Similarly we assume that \( u_k \) and \( y_k \) are bounded. It is further assumed that the upper bound of \( u_k \) is known and is called \( u_m \). The nonlinear functions \( b \) and \( f \) are approximated by neural networks, i.e.

\[
b(y_k, y_{k-1}, \ldots, y_{k-n}) = \sum_{l=0}^{M_1} g_{1l}(y_k, y_{k-1}, \ldots, y_{k-n})w_{1l} + \epsilon_{e1}
\]

\[
f(y_k, y_{k-1}, \ldots, y_{k-n}) = \sum_{l=0}^{M_2} g_{2l}(y_k, y_{k-1}, \ldots, y_{k-n})w_{2l} + \epsilon_{e2}
\]

where \( \epsilon_{e1}, \epsilon_{e2} \) satisfy \( |\epsilon_{e1}| \leq \epsilon_{m1}, |\epsilon_{e2}| \leq \epsilon_{m2} \) and \( \epsilon_{m1}, \epsilon_{m2} \) are the neural network approximation accuracy. The neural network weights are unknown. They are the desired weights.

The Equation 2 is then rewritten as

\[
y_{k+1} = \sum_{l=0}^{M_1} g_{1l}w_{1l}u_k + \sum_{l=0}^{M_2} g_{2l}w_{2l} + \epsilon_{ek}
\]

where \( \epsilon_{ek} = u_k\epsilon_{e1} + \epsilon_{e2} \) and \( |\epsilon_{ek}| \leq \epsilon_{m1}u_m\epsilon_{m1} + \epsilon_{m2} \).

This analysis states that two neural networks may be capable of predicting Equation 2. So the following prediction model is employed

\[
y_{k+1} = u_k \sum_{l=0}^{M_1} g_{1l}\hat{w}_{1l}^k + \sum_{l=0}^{M_2} g_{2l}\hat{w}_{2l}^k
\]

where \( \hat{w}_{1l}^k, \hat{w}_{2l}^k \) are the actual neural network weights. The following theorem guarantees stable prediction results.

**Theorem 2:** Provided \( u_k \) and \( y_k \) are bounded, the following neural network learning algorithms guarantee that the neural network weights will approach fixed bounded values
and \( \lim_{k \to \infty} |y_k - \hat{y}_k| \leq e_m \frac{2e_m}{\beta - \alpha}. \)

Define

\[
y_{k+1} = y_{k+1} - \hat{y}_{k+1}, \quad c_k = 1 + u_k \sum_{i=0}^{M_1} \delta_{k+1} + \sum_{i=0}^{M_2} \delta_{k+i}^2, \quad e_{\Delta k} = \begin{cases} \hat{y}_{k+1} & \text{if } |\hat{y}_{k+1}| > e_m \\ 0 & \text{if } |\hat{y}_{k+1}| \leq e_m \end{cases}
\]

Neural network learning algorithms

\[
\hat{w}_{li}^{k+1} = \hat{w}_{li}^k + \frac{\alpha}{c_k} u_k g_{i}(e_{\Delta k}) \quad i=0,1,\ldots,M_1 \tag{23}
\]

\[
\hat{w}_{li}^{k+1} = \hat{w}_{li}^k + \frac{\alpha}{c_k} g_{li}(e_{\Delta k}) \quad i=0,1,\ldots,M_2 \tag{24}
\]

where \( \alpha, \beta \) are constants and \( 0 < \alpha < \beta < 2 \), \( \alpha \) is called the learning rate.

**Remark 4:** The final prediction error depends on the neural network approximation errors and the upper bound of control signal \( u_k \). It can still be as small as required by increasing the neural network approximation accuracy.

**Example 2:** The system is \( y_{k+1} = \sin(0.5 \pi y_k)(\sin(\pi y_{k-1}))^2 + \sin(y_k - 0.5)u_k \). Comparing with Equation 2, we have \( f = \sin(0.5 \pi y_k) \sin(\pi y_{k-1})^2 \) and \( b = \sin(y_k - 0.5) \). The control input is \( \sin(\pi k/25) \). We know that the system output is in range \([-2,2]\).

Two RBF neural networks with Gaussian nodes are used. The first one is used to approximate nonlinear function \( f \). It has two inputs \( y_k \) and \( y_{k-1} \). The centres of its Gaussian nodes locate on a grid with distance 0.1 in each axis.

\( \sigma^2 = 39.27 \). Since \( y_k \) is in \([-2,2]\), only the nodes whose centres are in \([-2.5,2.5] \times [-2.5,2.5]\) are used. It has 2601 nodes.

Figure 5: System Output (solid line) and Its Prediction (dashed line). (a) The First 200 Points, (b) The Last 200 Points.
in total. The second neural network is used to approximate $b$. It has only one input $y_t$. The centres of its Gaussian nodes locate on a grid with distance 0.05 in the input axis. $\sigma^2 = 157.08$. The nodes whose centres are in the range $[-2.5, 2.5]$ are used. It has 101 nodes in total. 2,000 points are simulated. The learning rate is chosen to be 0.5. The deadzone is 0.01 (see Remark 3). The first 200 simulation points are shown in Figure 5a and the last 200 simulation points are shown in Figure 5b. The solid line is the system output. The dashed line is the prediction. After 2,000 points the prediction perfectly matches the system output.

4. CONTROL OF NONLINEAR DISCRETE SYSTEMS

In neural control, the neural network output is the plant input while the plant output is used in training and, therefore, affects the neural network output. A serious question here is, "Will these two coupled systems work in a converging fashion?". This question is solved by a stability guarantee. We shall call the error between the desired output and the actual system output the control error.

4.1 Nonlinear Systems of Class 1

This subsection discusses the adaptive control of systems described by Equation 1, where $f$ is an unknown nonlinear function, and $a_i$ and $b_i$ are unknown parameters. Narendra, 1992, presented a neural adaptive controller for this class of systems where $b_0$ is known and all roots of equation $b_0 + b_1 z + \ldots + b_m z^m = 0$ lie out of the unit circle. Jin et al., 1992, presented a more general adaptive control structure based on hyperstability theorems when the delay is 1 ($d = 0$ in Equation 1). In this subsection we extend Narendra's results to the case where $b_0$ is unknown. Neural network approximation error is also considered. The result is an indirect control method. The plant dynamics are first estimated using a method similar to the one used in prediction. The control effort is then calculated based on the estimated dynamics. It is assumed that;

**Assumption 1**: A closed compact set which $b_0$ belongs to is known and the set does not include 0. Without loss of generality we assume $0 < c_0 \leq b_0 \leq c_1$ where $c_0$ and $c_1$ are two known constants.

**Assumption 2**: Zeros of $b_0 + b_1 z + \ldots + b_m z^m$ are out of the unit circle.

**Assumption 3**: The nonlinear function $f$ satisfies a global Lipschitz condition.

Let nonlinear function $f$ be approximated by a neural network like Subsection 3.1. The same estimation model as Equation 11 is used. However the result is different since
$u_k$ and $y_k$ are no longer guaranteed to be bounded and are dependent on the controller design. This new result is presented in the following theorem.

**THEOREM 3**: The following neural network learning algorithm and linear parameter $(a_i, b_i)$ estimation algorithms yield the result that the error between the actual system output and the estimated output will either be within the bound $e_m$ or grow slower than the actual system outputs and control signals, i.e. $y_{k+1} - \hat{y}_{k+1} = o(\|\phi_k\|)$ where $\phi_k = (y_k, \ldots, y_{k-n}, u_{k-d}, \ldots, u_{k-d-n}, \theta_0, \ldots, \theta_M)^T$. The neural network weights and estimations of $a_i$ and $b_i$ will also approach fixed bounded values.

The neural network learning algorithm

$$\hat{w}_{i+1} = \hat{w}_i - \frac{\alpha}{c_k} e_{\Delta t} \quad i=0,1,\ldots,M$$  \hspace{1cm} (25)

The $a_i$ estimation algorithm

$$\hat{a}_{i+1} = \hat{a}_i + \frac{\alpha}{c_k} y_{k-d} e_{\Delta t} \quad i=0,1,\ldots,n$$  \hspace{1cm} (26)

The $b_i$ estimation algorithm

$$\hat{b}_{i+1} = \hat{b}_i + \frac{\alpha}{c_k} u_{k-d} e_{\Delta t} \quad i=1,\ldots,m$$ \hspace{1cm} (27)

$$(\hat{b}_{0+1})' = \hat{b}_0 + \frac{\alpha}{c_k} u_{k-d} e_{\Delta t}$$ \hspace{1cm} (28)

$$\hat{b}_{0+1} = \begin{cases} c_0 & \text{if } (\hat{b}_{0+1})' \leq c_0 \\ (\hat{b}_{0+1})' & \text{if } c_0 < (\hat{b}_{0+1})' < c_1 \\ c_1 & \text{if } (\hat{b}_{0+1})' \geq c_1 \end{cases}$$ \hspace{1cm} (29)

where $c_k$, $e_{\Delta t}$, $\alpha$ are the same as ones in Theorem 1.

**Remark 5**: The algorithm presented in the theorem is a standard gradient descent method with projection. The neural network learning algorithm and the $a_i$, $b_i$ estimation algorithms are the same as those in Subsection 3.1, except $b_0$ whose estimated values are projected to the known closed compact set if they are out of the set.

Now we consider calculating the control effort based on the estimated model. The same procedure as Narendra's (1992) is used. First consecutive estimations of $y$ from the current point $k$ are carried out, i.e.
\[
\begin{align*}
\tilde{y}_j^k &= y_j & \text{if } j \leq k \\
\tilde{y}_{j+1}^k &= \sum_{i=0}^{n} a_i \tilde{y}_{k-i}^k + \sum_{i=1}^{m} b_i \tilde{u}_{k-i}^k + \sum_{i=0}^{M} g_i(\tilde{y}_j^k, \ldots, \tilde{y}_{j-i}^k) \tilde{w}_i^k & \text{if } j \geq k
\end{align*}
\] (30)

Then the control signal is calculated as
\[
\begin{align*}
u_k &= (\beta_0^*)^{-1}(y_{k+1} - \sum_{i=1}^{n} a_i \tilde{y}_{k-i}^k - \sum_{i=1}^{m} b_i \tilde{u}_{k-i}^k - \sum_{i=0}^{M} g_i(\tilde{y}_k^k, \ldots, \tilde{y}_{k-i}^k) \tilde{w}_i^k)
\end{align*}
\] (31)

**Theorem 4:** The control algorithms (Equation 30 and Equation 31) guarantee that the control signal and the system output are bounded and that the control error is bounded. For the same neural network structure, the control error reduces with increasing of neural network approximation accuracy.

**Remark 6:** The theorem only establishes the boundness of the control error. The bound limit is not only related to the neural network approximation accuracy but also related to the estimations of parameters and neural network weights.

**Remark 7:** The control error is directly related to both approximation accuracy, \(e_m\), and neural network structure. If we attempt to improve \(e_m\) by network reconstruction we may not improve the overall control error. However simulations show that reconstruction does improve the situation.

**Example 3:** The example 11 in Narendra, 1992, (i.e. \(y_{k+1} = 0.9y_k + 0.3y_{k-1} + 0.3y_{k-2} + 1.2\sin(y_k) + 0.6\cos(y_{k-1}+y_{k-2}) + y_{k-2} + 0.2u_{k-1} + 0.16u_{k-2}\)) is used here but this time the \(b_0\) is unknown. We only know that \(b_0\) is in range \([0.1, 0.4]\). The same neural network as the one in Narendra, 1992, is used.

The deadzone is chosen as 0.01 (see Remark 3). 20,000 points in total are simulated. The desired output and the system output in the first and last 200 points are compared in Figure 6.

\[\text{Figure 6: The Desired Output (solid line) and The System Output (dashed line). (a) The First 200 Points, (b) The Last 200 Points.}\]
The solid line is the desired output and the dashed line is the system output. The system output and the estimation output are compared in Figure 7. Here the solid line is the system output and the dashed line is the estimation output. The control effort in the first and last 200 points is shown in Figure 8. There have quite big control signals in the first 200 points. However the control signals stay small in the last 200 points. Figure 6-8 are comparing to Figure 5.22-5.24 in Narendra, 1992.

4.2 Nonlinear Systems of Class 2

In this subsection we discuss the adaptive control of systems described by Equation 2, where $b$ and $f$ are two unknown nonlinear functions. The result is a direct control method where no estimation model is used. It is assumed that

**Assumption 4**: The sign of the nonlinear function $b$ is known.

**Assumption 5**: $b$ is bounded. The bound limit does not cross 0. Without loss of generality we assume $0 < b_0 \leq b \leq b_1$, where $b_0$ and $b_1$ are two constants and $b_1$ is known.
Assumption 6: $y^*_k$ is the desired system output and its values are bounded. The upper bound is known as $y^*_m$.

From Equation 2 we have

$$b^{-1}y_{k+1} = b^{-1}f + u_k$$

(32)

or

$$b^{-1}y_{k+1} = b^{-1}y_{k+1}^* - b^{-1}f - u_k$$

(33)

where $y_k = y_k^* - y_k^*$.

Now we let the nonlinear functions $b^{-1}$ and $b^{-1}f$ be approximated by neural networks, i.e.

$$b^{-1} = \sum_{i=0}^{M_1} g_{1i}(y_k, \ldots, y_{k-1})w_{1i} + \epsilon_{ek1}$$

(34)

$$b^{-1}f = \sum_{i=0}^{M_2} g_{2i}(y_k, \ldots, y_{k-1})w_{2i} + \epsilon_{ek2}$$

(35)

where $\epsilon_{ek1}, \epsilon_{ek2}$ satisfy $|\epsilon_{ek1}| \leq \epsilon_{m1}, |\epsilon_{ek2}| \leq \epsilon_{m2}$ and $\epsilon_{m1}, \epsilon_{m2}$ are the neural network approximation accuracy. Substituting $b^{-1}$ and $b^{-1}f$ in Equations 33 by Equation 34 and Equation 35, we have

$$b^{-1}y_{k+1} = y_{k+1}^* \sum_{i=0}^{M_1} g_{1i}w_{1i} - \sum_{i=0}^{M_2} g_{2i}w_{2i} - u_k + \epsilon_{ek}$$

(36)

where $\epsilon_{ek} = y_{k+1}^* \epsilon_{ek1} - \epsilon_{ek2}$ and $|\epsilon_{ek}| \leq \epsilon_m y^*_m + \epsilon_{m2} \Delta \epsilon_m$.

It is reasonable to design the controller as $u_k = y_{k+1}^* \sum_{i=0}^{M_1} g_{1i}\hat{w}_{1i}^k - \sum_{i=0}^{M_2} g_{2i}\hat{w}_{2i}^k$ where $\hat{w}_{1i}^k, \hat{w}_{2i}^k$ are the actual neural network weights. The control diagram is shown in Figure 9. The following neural network learning algorithms guarantee stable results.

**THEOREM 5:** Provided the assumptions are satisfied, the following neural network learning algorithms guarantee that the neural network weights approach fixed bounded values and that the system output approaches the desired output within the error

$$\epsilon_m = \frac{2b_1 \epsilon_m}{\beta - db_1}$$

where $d$ and $\beta$ are two constants, defined as follows

Define
\[ 0 < d < \beta/b, \ 0 < \beta < 2, \ c_k = d^2 + \left( \sum_{i=0}^{M_i} g_i \right)^2 + \sum_{i=0}^{M_i} g_{2i}^2 \]

\[ e_{\Delta} = \begin{cases} \gamma_{k+1} & \text{if } |\gamma_{k+1}| > e_m \\ 0 & \text{if } |\gamma_{k+1}| \leq e_m \end{cases} \]

Neural network learning algorithms

\[ \hat{w}_{li}^{k+1} = \hat{w}_{li}^{k} + \frac{d}{c_k} g_{li} \gamma_{k+1} \quad (37) \]

\[ \hat{w}_{2i}^{k+1} = \hat{w}_{2i}^{k} - \frac{d}{c_k} g_{2i} e_{\Delta} \quad (38) \]

**Remark 8:** The control error depends on the neural network approximation accuracy and the upper bounds of \( b \) and \( y_k \). Since the upper bounds are fixed, the control error can be made as small as required by increasing neural network approximation accuracy through changing neural network structures.

**Example 4:** The system is described by the difference equation

\[ y_{k+1} = \frac{1}{8} \frac{\sin(4 \pi y_k)}{\pi y_k} \left( \frac{\sin(\pi y_{k-1})}{\pi y_{k-1}} \right)^2 + (2 + \sin(3 \pi (y_k - 0.5))) u_k \]

Comparing with Equation 2 we have

\[ f = \frac{1}{8} \frac{\sin(4 \pi y_k)}{\pi y_k} \left( \frac{\sin(\pi y_{k-1})}{\pi y_{k-1}} \right)^2 \]

\[ b = b(y_k) = 2 + \sin(3 \pi (y_k - 0.5)) \]

The functions \( f \) and \( b \) are unknown, but we know that the sign of \( b \) is positive, and \( b \) is bounded away from zero and bounded above by 4. The desired system output is

\[ y^*_k = 0.343 r_k + 0.9 y_{k-1}^* - 0.27 y_{k-2}^* + 0.027 y_{k-3}^* \]

where \( r_k \) is a unit square wave with period = 100.

Two Gaussian neural networks are built to approximate \( b^1 f \) and \( b^1 \). The interesting domains of \( y_k \) and \( y_{k-1} \) are \([-1, 2]\). The neural network for \( b^1 \) has one input \( y_k \). The
centres of its Gaussian nodes locate on a grid with distance 0.02 in its input axis. $\sigma^2 = 986.96$. Only nodes whose centres are in $[-1.12, 2.12]$ are used. There are 163 nodes in total. The neural network for $b^i f$ has two inputs $y_k$ and $y_{k-1}$. The nodes of this neural network have the form $e^{-(y_k - \xi_k)^2 + (y_{k-1} - \xi_{k-1})^2}$.

The centres of $y_k$ (i.e. $\xi_k$) locate in a grid with distance 0.04 in its input axis (i.e. $y_k$). The centres of $y_{k-1}$ (i.e. $\xi_{k-1}$) locate on a grid with distance 0.15 in its input axis ($y_{k-1}$). $\sigma^2 = 246.74$. $\sigma^2 = 19.74$. There are 3234 nodes in total.

$d$ is chosen to be $1.5 / b_1 = 0.375$. The deadzone is chosen to be 0.01 (see Remark 3). 1000 points are simulated. Figure 10 shows the desired output and the system output in the first and last 200 points. The solid line is the desired output and the dashed line is the system output. The control efforts in the first and last 200 points are shown in Figure 11.

![Figure 10](image1.png)  
**Figure 10:** The Desired Output (solid line) and The System Output (dashed line). (a) The First 200 Points, (b) The Last 200 Points.

![Figure 11](image2.png)  
**Figure 11:** The Control Efforts. (a) The First 200 Points, (b) The Last 200 Points.

5. CONCLUSIONS

This paper presents how to use neural networks to predict and control two classes of
discrete nonlinear systems. Gradient estimation with deadzone is used for developing the neural network learning algorithms and the linear parameter estimation algorithms. The dead beat method is used to design the controller. All results have strict theoretical bases. Examples are employed to show how the theoretical results work. Simulations show that the neural network control and prediction work very well even though the neural networks have not been initially trained. However initial off-line training from prior knowledge should improve the on-line performance. For example, off-line training can eliminate the early vibrations in example 4. It can also decrease the large initial control output in example 3.

As pointed out in example 1, there is a difficulty in applying the theoretical results, i.e. difficult to calculate the deadzone. The deadzone is determined by the neural network structure and the nonlinear function which the neural network is used to approximate. Therefore, to calculate the deadzone requires plant knowledge. This difficulty is solved by trial and error method in all examples. Another possible solution may be to use a self exciting control method such as that found in Giri et al., 1992.

Example 3 shows that the system output follows the desired output very slowly. However the estimation output follows the system output very quickly. The problem therefore comes from the controller design. A new control method should be investigated.

Assumption 2 limits the results in Subsection 4.1 only suitable to minimum phase systems. In order to extend the results into nonminimum phase systems, the pole placement control design method should be used. When the constrained set of parameters is known and within this set all possible systems are stabilizable, gradient estimation may be used, e.g. Wen & Hill, 1989. Otherwise Least Square estimation, e.g. Giri et al., 1992, is needed. However this will make the neural networks much more complicated. One possible solution is to estimate the LS method as people do in the field of high order neural networks, e.g. Moller, 1990, and Battiti, 1991.

REFERENCES


APPENDIX A

Basic Theorem: Given a system

\[ y_{k+1} = \phi_k^T \theta_k + e_k, \quad |e_k| \leq e_m \]  

(A.1)

The parameter vector \( \theta_k \) is adjusted as follows

\[ \theta_{k+1} = \theta_k - a_k \frac{\phi_k e_{\Delta t}}{1 + \phi_k^T \phi_k} \]  

(A.2)

\[ e_{\Delta t} = \begin{cases} y_{k+1} & \text{if } |y_{k+1}| > e_k \\ 0 & \text{if } |y_{k+1}| \leq e_k \end{cases} \]  

(A.3)

where \( 0 < a_0 \leq a_k < \beta, 0 < \beta < 2 \), \( \beta \) and \( a_0 \) are two constants, \( e_k \) is not necessarily a constant but must have \( e_k \geq \frac{2e_m}{\beta - a_k} \). Provided that vector \( \phi_k \) is bounded or

\[ \|\phi_k\| \leq c_1 + c_2 \max_{0 \leq i \leq k} |y_i| \]  

where \( c_1 \) and \( c_2 \) are constants, \( y_{k+1} \) will approach 0 with the error not greater than \( e_k \) and the parameters will approach fixed bounded values.

Proof:

From Equation A.2 we have

\[ \| \theta_{k+1} \|^2 - \| \theta_k \|^2 = \frac{a_k}{1 + \phi_k^T \phi_k} [2(y_{k+1} - e_k) e_{\Delta t} - a_k e_{\Delta t} e_{\Delta t} - \phi_k^T \phi_k - \phi_k^T \phi_k] \]  

(A.4)

If \( |y_{k+1}| \leq e_k \), the right hand of Equation A.4 is 0.

If \( |y_{k+1}| > e_k \), the contents in the square brackets are

\[ [(2 - \beta)y_{k+1}^2] + [((\beta - a_k)y_{k+1}^2 - 2e_k y_{k+1})] + [\frac{a_k y_{k+1}^2}{1 + \phi_k^T \phi_k}] \]  

(A.5)

The contents of the all three pairs of square brackets are non-negative.

In either case we prove that \( \| \theta_k \| \) is a non-increasing scalar. In addition it is
bounded below by 0 so it must converge to a fixed value. Therefore we have that $\theta_k$ is bounded.

Now we prove that in either case $e_{\Delta k}$ approaches 0 and the parameters approach fixed values. In the former case ($|y_{k+1}| \leq e_k$) we already have the results. In the latter case ($|y_{k+1}| > e_k$), since $\|\theta_k\|$ approaches a fixed value, we have

$$\lim_{k \to \infty} (\|\theta_{k+1}\|^2 - \|\theta_k\|^2) = 0.$$  

This results in

$$\lim_{k \to \infty} \frac{a_k(2-\beta)y_{k+1}^2}{1 + \phi_k^T \phi_k} = 0 \quad \text{(A.6)}$$

Following the proof of Lemma 3.3.2 in Goodwin, 1984, we have that the parameters approach fixed values. From Equation A.6, the theorem conditions, and Lemma 3.1 in Goodwin et al., 1980, we have $\lim y_k = 0$ which also gives $e_{\Delta k} = 0$. $\lim e_{\Delta k} = 0$ is the same as $\lim_{k \to \infty} |y_{k+1}| = e_k$.

Remark A.1: The boundness of parameters and their approach to fixed values do not use the boundness of $\phi_k$ or $\|\phi_k\| \leq c_1 + c_2 \max_{0 \leq i \leq k} |y_i|$.

APPENDIX B

Proof of Theorem 1:

Subtracting Equation 11 from Equation 10 we have

$$y_{k+1} = \phi_k^T \bar{\phi}_k + e_{\Delta k} \quad \text{(A.7)}$$

where

$$\bar{\theta}_k = (a_0 - \tilde{a}_k^0, \ldots, a_n - \tilde{a}_n^k, b_0 - \tilde{b}_0^k, \ldots, b_m - \tilde{b}_m^k, w_0 - \tilde{w}_0^k, \ldots, \tilde{w}_r - \tilde{w}_r^k)^T,$$

$$\phi_k = (y_k, \ldots, y_{k-a}, u_{k-d}, \ldots, u_{k-d-m}, 8_0, \ldots, 8_m)^T.$$

The neural network learning algorithm and parameter estimation algorithms are equivalent to

$$\bar{\theta}_{k+1} = \bar{\theta}_k - \frac{\alpha \phi_k^T e_{\Delta k}}{1 + \phi_k^T \phi_k} \quad \text{(A.8)}$$

Since $u_k$, $y_k$ are bounded and the nonlinear mapping vector of the neural network is
bounded as well, the vector $\phi_k$ is bounded. The theorem result directly comes from the Basic Theorem.

APPENDIX C

Proof of Theorem 2:

From Equation 21 and Equation 22, we have

$$y_{k+1} = g_k^T \tilde{W}_k + \epsilon_{sk}$$  \hspace{1cm} (A.9)

where $g_k = (u_k g_1, \ldots, u_k g_{2M}, \delta_{20}, \ldots, \delta_{2M})^T$,

$$\tilde{W}_k = (\tilde{w}_{10}, \ldots, \tilde{w}_{1M}, \tilde{w}_{20}, \ldots, \tilde{w}_{2M})^T.$$  

The neural network learning algorithms are equivalent to

$$\tilde{W}_{k+1} = \tilde{W}_k - \alpha \frac{g_k e_{sk}}{1 + g_k^T g_k}$$  \hspace{1cm} (A.10)

Since $u_k$ is bounded and the neural network nonlinear mapping vectors are bounded, then $g_k$ is bounded. Using the Basic Theorem, we have the result of Theorem 2.

APPENDIX D

Proof of Theorem 3:

Define $\theta' = (a_0 - \hat{a}_0, \ldots, a_n - \hat{a}_n, b_0 - (\hat{b}_0)' \ldots, b_m - \hat{b}_m, w_0 - \hat{w}_0, \ldots, w_M - \hat{w}_M)^T$ and $\tilde{\theta}_k$ is the same as in Appendix B. The parameter adjusting algorithm from $\theta_k$ to $\tilde{\theta}_{k+1} = \tilde{\theta}_k$ is the same as Equation A.2. The algorithm from $\tilde{\theta}_{k+1}$ to $\tilde{\theta}_{k+1}$ is projection. Therefore we have

$$\| \tilde{\theta}_{k+1} \| \leq \| \tilde{\theta}'_{k+1} \| \leq \| \tilde{\theta}_{k} \|$$  \hspace{1cm} (A.11)

This is equivalent to saying that $\| \tilde{\theta}_k \|$ approaches a fixed value so $\lim_{k \to \infty} (\| \tilde{\theta}'_{k+1} \| - \| \tilde{\theta}_k \|) = 0$. Following the proof of the Basic Theorem, we have that the neural network weights and estimations of $a_i$, $b_i$ approach fixed bounded values. Also we have either $e_{sk} = 0$ or $\lim_{k \to \infty} \frac{\tilde{y}_{k+1}^2}{1 + \phi_k^T \phi_k} = 0$. Therefore we have either $|y_k| \leq e_m$ or
\[ J_{k+1} = o(\| \phi_k \|) \] when k approaches infinity.

**APPENDIX E**

**Proof of theorem 4:**

Combining the control equations 30-31 and the estimation equation 11, we have

\[
y_{k+d} - \beta_k y_{k+d}^* = \sum_{i=0}^{n} (\alpha_i^{k+d} - \beta_i \alpha_i^k) y_{k-i} + \beta_k \sum_{i=0}^{n} \alpha_i^k (y_{k+1-i} - \bar{y}_{k+1-i}) \\
+ \frac{m^k}{\beta_k} \sum_{i=0}^{M} g_i(y_{k+1-d}, \ldots, y_{k+d}) (\bar{y}_{k+d-i} - \beta_k \bar{y}_i)
\]

where \( \beta_k = b_0^{k+d} / b_0^{k} \).

When k approaches infinity, we have (from Theorem 3)

\[
\lim_{k \to \infty} \beta_k = 1, \quad \lim_{k \to \infty} (\alpha_i^{k+d} - \beta_i \alpha_i^k) = 0, \quad \lim_{k \to \infty} (\bar{y}_{k+d-i} - \beta_k \bar{y}_i) = 0
\]

Now we prove that \( y_k \) is bounded. If it is unbounded, from the discussions in Narendra 1992, we have \( u_k = O(y_{k+d}) \) and \( y_{k+d-i} - \bar{y}_{k+d-i} = o(y_{k+d}) \). In addition, the neural network nonlinear mapping vector, the neural network weights, and the desired output are bounded so, from Equation A.12, we have \( \bar{y}_{k+d+1} = o(y_{k+d+1}) \). Theorem 3 provides that \( y_{k+d+1} - \bar{y}_{k+d+1} = o(y_{k+d+1}) \). Therefore we have

\[
y_{k+d+1} = o(y_{k+d+1})
\]

Equation A.13 contradicts the unboundedness of \( y_k \), so the system output must be bounded.

Since the system output is bounded, the control signals are also bounded. The control error can be expressed as

\[
\lim_{k \to \infty} |y_{k+d} - y_{k+d}^*| \leq \lim_{k \to \infty} |y_{k+d} - \bar{y}_{k+d}| + \lim_{k \to \infty} |\bar{y}_{k+d} - y_{k+d}^*| \\
\leq e_m + \lim_{k \to \infty} |\sum_{i=0}^{n} \alpha_i^k (y_{k+1-i} - \bar{y}_{k+1-i})| \\
\leq \lim_{k \to \infty} \left| \sum_{i=0}^{M} (g_i(y_{k+1-d}, \ldots, y_{k+d}) - g_i(\bar{y}_{k+1-d}, \ldots, \bar{y}_{k+d})) \bar{y}_i \right|
\]

Therefore the control error is bounded. Furthermore \( y_{k+d-i} - \bar{y}_{k+d-i} \) are functions of the
neural network approximation accuracy and will decrease when the accuracy increases. We have the theorem results.

**APPENDIX F**

**Proof of Theorem 5:**

We combine the neural network learning algorithms of Equations 37 and 38

\[
\tilde{W}^{k+1} = \tilde{W}^k - \frac{d \ast g_k e_{ak}}{d^2 + g_k^T g_k}
\]  
(A.15)

where \( \tilde{W}^k = (\tilde{w}_{10}^k, \ldots, \tilde{w}_{1M}^k, \ldots, \tilde{w}_{o1}^k, \ldots, \tilde{w}_{o2M}^k)^T \),

\( g_k = (g_{10}^k, g_{1M}^k, \ldots, g_{o1}^k, \ldots, g_{o2M}^k)^T \).

Or

\[
\tilde{W}^{k+1} = \tilde{W}^k - \frac{d \ast b \ast \left( \frac{g_k}{d} \right) e_{ak}}{1 + \left( \frac{g_k}{d} \right)^T \left( \frac{g_k}{d} \right)}
\]  
(A.16)

Dividing both sides of Equation 36 by d, we have

\[
\frac{1}{d \ast b} \tilde{y}_{k+1} = \tilde{W}^r g_k + \frac{e_{ak}}{d}
\]  
(A.17)

Now we consider Equation A.16 as the parameter adjustment algorithm of system of Equation A.17. Before applying the basic theorem results, we analyze the learning rate, the size of deadzone and the boundness of \( g_k / d \).

Since \( d \) is chosen to be a constant with \( 0 < d < \beta / b_1 \), \( d \ast b \) is in the range \( 0 < d \ast b_0 \leq d \ast b < \beta \). The learning rate satisfies the basic theorem condition. The approximation error in Equation A.17 is \( e_{ak} / d \) which satisfies \( |e_{ak} / d| \leq e_m / d \). The deadzone in Equation A.16 is \( \frac{2b_1 e_m}{\beta - d \ast b_1} \frac{1}{d \ast b} \geq \frac{2e_m}{\beta - d \ast b} \) so the deadzone also satisfies the basic theorem condition. Since the desired output is bounded and the nonlinear mapping vectors of the neural network are also bounded, \( g_k / d \) is bounded. Therefore all the conditions of the basic theorem are satisfied. Using the basic theorem results we have that the neural network weights approach fixed bounded values and the output of Equation A.17 approaches the deadzone or

\[
\lim_{k \to \infty} \tilde{y}_k \leq \frac{2b_1 e_m}{\beta - d b_1}.
\]