A self growing binary tree neural network for the identification of multiclass systems

by

K. M. Tsang[*]

and

S. A. Billings[†]

[*]Department of Electrical Engineering
Hong Kong Polytechnic
Hung Hom
Kowloon
Hong Kong

[†]Department of Automatic Control and Systems Engineering
University of Sheffield
Sheffield S1 3JD
UK

Research Report No 451

March 1992

Abstract

A self growing binary tree neural network is introduced for the on-line identification of multi-class systems. Instead of solving the traditional two class problem with a single neuron a time shrinking threshold logic unit is introduced such that the modified neuron has the capability of partitioning the raw data records into three different regions. Incorporating a Least Mean Squares (LMS) learning algorithm provides the capability of detecting and creating new classes and this allows clustering and partitioning of the raw data records into model classes and yields estimates of the parameters. The models which describe the behaviour of the system at different operating regions can be recovered by inspection of the connection weights of the individual neurons. Optimisation procedures for on-line estimation are also proposed. Simulation studies are included to illustrate the concepts.

1. Introduction

Multi-class systems which can be described by a class of models over different operating regions are often encountered in practice. Systems which contain piece-wise linear, hysteresis and coulomb friction elements are typical examples of multi-class systems. Fitting one global model to systems which include these elements may give unsatisfactory results and the final model may fail to capture the behaviour of the underlying system [1,2]. A stepwise backward elimination procedure based on an orthogonal least squares algorithm coupled with discriminant functions provides one way of identifying multi-class systems [1]. But, this is an off-line algorithm and a complete set of input and output data is required for the analysis. Operating the backward elimination procedure on-line is almost impossible and a more effective and efficient algorithm is required to cluster and partition the data records and to perform the parameter estimation. Correct classification or partitioning of the data into classes is vitally important in multi-class system identification because the classes that the incoming data records belong to, the decision surfaces, and the number of different classes of models describing the behaviour of the system are seldom known a priori.

Neural networks present a computational paradigm for constructing pattern recognition and learning algorithms. Early seminal works in the neural network areas include Rosenblatt [3], Widrow and Hoff [4], and Minsky and Papert [5]. A detailed summary of neural networks together with applications can be found in [6] and [7]. A perceptron [3] or an adaptive linear neuron [4] coupled with a threshold logic unit can form linear separable functions which can be used to partition the input pattern into regions. More elaborate classification functions can be constructed by using multi-layered neural networks [6,7]. The least mean squares (LMS) learning algorithm applied to the perceptron or the adaptive linear neuron is a simple and efficient algorithm which can easily be implemented on-line. An extension of the linear adaptive neuron to the on-line identification of multi-class systems by including procedures for partitioning the data records coupled with LMS parameter estimation would therefore appear to be worthwhile.

In the present paper a new self growing binary tree neural network is proposed based upon an extension of the adaptive linear neuron [4] for the identification of multi-class systems which can be described by a set of models over different operating regions. The proposed network exploits the properties of pattern classification and adaptation in the identification process. A simple linear neural network is used because the structure and parameters of the system under investigation can easily be recovered by inspection of the weight vectors of the network. Simulation studies are included to illustrate the results.

2. The adaptive linear neuron

A single adaptive linear neuron with m input signals and an output threshold logic element is shown in Figure 1. This unit has an input signal vector or input pattern vector $X(k) = [x_0(k)\ x_1(k)\ \ldots\ x_m(k)]^T$ whose components are weighted by a set of coefficients $W(k) = [w_0(k)\ w_1(k)\ \ldots\ w_m(k)]^T$. The unit produces an output

$$y(k) = X(k)^T W(k) \tag{1}$$

An adaptation algorithm automatically adjusts the weights $W(k)$ so that the output response to the input patterns will be as close as possible to the respective desired response. This is done by adjusting the weights using the LMS algorithm [4] so that

$$W(k+1) = W(k) + 2\mu\varepsilon(k)X(k) \tag{2}$$

where $\varepsilon(k)$ is the error between the desired response $y^*(k)$ and the output response from the neuron $y(k)$

$$\varepsilon(k) = y^*(k) - X(k)^T W(k) \tag{3}$$

and μ is the adaptation step. The choice of μ controls the stability and speed of convergence of the algorithm and a practical range of μ is [7]

$$0 < \mu < \frac{1}{trace[R]} \tag{4}$$

where $trace[R] = \sum (diagonal\ element\ of\ R)$ is the average signal of the X-vectors or $E[X(k)^T X(k)]$. With μ set within this range, the LMS algorithm should converge to W^* the optimal solution. A proof of this result can be found in [8]. The LMS algorithm minimises the sum of the squares of the linear errors over the training set.

During the training process, input patterns and the corresponding desired responses are presented to the linear neuron. The LMS algorithm automatically adjusts the weights so that the output response to the input pattern will be as close as possible to the respective desired response. Decisions are made by the threshold logic element shown in Figure 2 where the output response of the threshold logic element is governed by the error between the desired output and the output of the neuron.

2.1. The learning rule and partitioning of input patterns

In classification problems, the threshold logic element shown in Fig. 1 is normally selected to be a hard-limiting quantizer. This produces a binary ± 1 output to solve the standard two-class problem. That is

$$\begin{aligned} \epsilon(k) > 0 &\rightarrow X(k) \in \Omega_+ \\ \epsilon(k) < 0 &\rightarrow X(k) \in \Omega_- \end{aligned} \tag{5}$$

where $\epsilon(k)$ is the error between the desired output and the neuron output and $\Omega_+, \Omega_- \subset \Re^m$ define the two regions which the input patterns belonged to. In the identification of multi-class systems, the capability of the threshold logic unit is extended to cope with the three-class problem. Consider the partitioning of the input patterns according to the threshold logic unit shown in Figure 2. Three classes are partitioned according to the rule

$$\begin{aligned} |\epsilon(k)| &\leq \gamma(k) \rightarrow X(k) \in \Omega_0 \\ \epsilon(k) &> \gamma(k) \rightarrow X(k) \in \Omega_+ \\ \epsilon(k) &< -\gamma(k) \rightarrow X(k) \in \Omega_- \end{aligned} \tag{6}$$

where $\gamma(k)$ is the tolerance band of the threshold logic unit and Ω_0, Ω_+ and $\Omega_- \subset \Re^m$ define the three regions in the hyperplane that the input patterns belong to. If $\gamma(k) = 0$, the threshold logic unit in Fig.2 reduces to the two level hard limiter and the adaptive linear neuron described in Figure 1 becomes the standard perceptron or the standard adaptive linear neuron.

According to the partition rule of eqn.(6), the weight vector for the neuron will only be updated using the LMS algorithm if the error $\epsilon(k)$ falls within the tolerance band $(X(k) \in \Omega_0)$. If this condition is satisfied then

$$W(k+1) = W(k) + 2\mu\epsilon(k)X(k) \quad \forall \quad |\epsilon(k)| \leq \gamma(k), \; X(k) \in \Omega_0 \tag{7}$$

If $\epsilon(k)$ falls outside the tolerance band ($|\epsilon(k)| > \gamma(k)$ and $X(k) \notin \Omega_0$), the input pattern $X(k)$ will be excluded from the estimation and passed to a second layer neuron for further adaptation.

2.2. Asymptotic properties of the learning rule

For a well defined and completely separable multi-class system, based on the classification rule of eqn.(6), the weight vector $W(k)$ for the adaptive neuron is updated according to the LMS algorithm of eqn.(7) only if the input data pattern $X(k)$ belongs to Ω_0. Now consider the case when the tolerance limit $\gamma(k)$ is a time shrinking function

$$\gamma(k-1) > \gamma(k) \geq 0 \tag{8}$$

so that asymptotically the tolerance limit reduces to a very small value

$$\lim_{k \to \infty} \gamma(k) \to \delta \tag{9}$$

where δ is a small positive constant. At steady state, the tolerance limit can be approximated by

$$\lim_{k \to \infty} \gamma(k) \approx 0 \tag{10}$$

and according to the partition rule of eqn.(6) and learning rule of eqn.(7), the weight vector $W(k)$ will only be updated if the error $\epsilon(k)$ continues to fall within the tolerance band $\gamma(k)$ ($|\epsilon(k)| \leq \gamma(k)$). Because of the time shrinking property of the tolerance function $\gamma(k)$ the weight vector $W(k)$ should be slowly directed towards the true weight vector W^* which describes the behaviour of the data patterns $X(k)$ belonging to Ω_0. At steady state, the tolerance function will approach zero implying that the error for adaptation should also approach zero

$$\lim_{k \to \infty} \epsilon(k) \to 0 \quad \forall \quad X(k) \in \Omega_0 \tag{11}$$

Substituting eqn.(11) into eqn.(3) and taking the limit of $k \to \infty$ gives

$$y^*(k) = X(k)^T W^* \approx X(k)^T \lim_{k \to \infty} W(k) \tag{12}$$

That is with the time shrinking property of the tolerance limit $\gamma(k)$ and the learning rule of eqns.(6) and (7), the estimated weight vector $W(k)$ should converge to the true parameters which describe the behaviour of the data pattern belonging to Ω_0.

If there are N data patterns $X(k)$, $k=1,\dots,N$ classified as belonging to Ω_0 the misclassification rate for this set of data patterns can be defined as

$$J(N) = \frac{1}{N}\sum_{k=1}^{N} \epsilon^2(k) = \sum_{k=1}^{N} [y^*(k) - X(k)^T W(k)]^2 \tag{13}$$

From the partitioning and learning rules of eqns.(6) and (7), asymptotically $\gamma(k) \to 0$ and $W(k) \to W^*$ implies that $\lim_{k\to\infty} \epsilon(k) \to 0$ at the steady state. Hence for large values of N the misclassification rate of eqn.(13) should asymptotically converge to zero.

A plausible candidate for the tolerance limit $\gamma(k)$ in the region Ω_0 is

$$\gamma(k) = \beta^N \Gamma + \alpha \sqrt{J(N)} \quad , \quad k \geq N \tag{14}$$

at time k where N is the number of data patterns classified as belonging to Ω_0, β is a constant value with magnitude less than unity ($\beta < 1$), $J(N)$ is the misclassification rate for the activated neuron, Γ and α are some positive constants. Γ controls the initial training time of the neuron before the selection and classification process begins while α controls the asymptotic tolerance limit. A large value of Γ and a value of β close to unity would provide a long learning time for each neuron.

3. The binary tree neural net

More complicated classification functions can be implemented if the neuron shown in Fig.1 is stacked to form a multi-layered neural network. From the selection and partition rule of eqn.(6), two sets of data records have been excluded from the adaptation. They are

$$\epsilon(k) > \gamma(k) \quad \to \quad X(k) \in \Omega_+ \tag{15}$$

and

$$\epsilon(k) < -\gamma(k) \quad \to \quad X(k) \in \Omega_- \tag{16}$$

For data patterns satisfying either of these conditions a neuron on the second layer is "fired" or "activated" for adaptation. Data patterns $X(k) \notin \Omega_0$ are therefore passed on to second layer neurons for adaptation and the selection, partition and learning processes of eqns.(6) and (7) are implemented as before such that data patterns belonging to Ω_+ and Ω_- are further partitioned into three separate regions according to eqn.(6). This splitting process can carry on further to create a very complicated classification function and a binary tree neural network is formed. Termination of the binary tree is governed by the complexity of the incoming data patterns and the specified asymptotic tolerance band $\lim_{k\to\infty} \gamma(k)$. An example of a three layered binary tree network, which can classify at most seven (2^3-1) different patterns, is shown in Fig.3.

One special characteristic of the proposed network is that if the input patterns get very complicated and are composed of a number of different classes the binary tree neural net will automatically grow and find new links and weights for the newly connected neurons until all input patterns are classified into respective classes. This neural net is therefore a self growing network which has the capability of creating a new class by "firing" or "activating" a new neuron by making a connection to it. When a new pattern arrives a new neuron on the next layer will be "fired" for adaptation and a new link to the neuron will form provided the new pattern is well defined and can be separated from the rest of the existing classes of patterns. The classification, adaptation and self growing properties of the binary tree neural net are very important in multi-class system identification because the number of different classes of models describing the behaviour of the system and the partition planes for the incoming data patterns are seldom known a priori. The binary tree neural net provides one way of detecting the number of classes of models which describe the system as well as producing the partition planes and parameter estimates for these classes. In the worst possible case the binary tree neural network will make as many links to neurons as the number of patterns presented. A further advantage of this approach is that the linear structure of the network allows the system parameters and structure to be directly recovered from the links to the neurons.

4. Implementation of the binary tree neural net

The implementation of the binary tree neural net can be summarised as follows:-

i) Start with one neuron initially. Set all weight vectors $W(k)$ to zero. Define μ, $\gamma(k)$ and α. Set up the input patterns $X(k)$.

ii) Evaluate the error $\epsilon(k)$ for each pattern presented using eqn.(3).

iii) Perform the classification test on $\epsilon(k)$ using eqn.(6).

   a) If $|\epsilon(k)| \leq \gamma(k)$, classify $X(k)$ as belonging to Ω_0, the activated neuron and update the weight vector $W(k)$ according to eqn.(7).

   b) If $\epsilon(k) > \gamma(k)$, "fire" or "activate" a neuron on the next layer for adaptation. The data vector $X(k) \in \Omega_+$ is passed on to the new activated neuron for further classification and adaptation. Repeat procedures ii) and iii) until the data pattern $X(k)$ settles on a neuron.

c) If $\epsilon(k) < -\gamma(k)$, "fire" a neuron on the next layer for adaptation. The data vector $X(k) \in \Omega_-$ is passed on to the new activated neuron for further classification and adaptation. Repeat procedures ii) and iii) until the data pattern $X(k)$ settles on a neuron.

iv) Get a new data pattern. Repeat procedures ii) and iii).

5. Illustrated examples

The operation and effectiveness of the proposed binary tree neural net is best illustrated by examples.

5.1 A deterministic multi-class system

Consider a piece-wise linear system (S_1) described by the equation

$$x(k) = \begin{cases} 0.4u(k-1) + 0.5x(k-1) + 0.6 \ ; & u(k-1) > 1 \\ u(k-1) + 0.5x(k-1) \ ; & |u(k-1)| \le 1 \\ 0.4u(k-1) + 0.5x(k-1) - 0.6 \ ; & u(k-1) < -1 \end{cases} \tag{17}$$

where $u(k)$ and $x(k)$ are the input and output of the system respectively. A zero mean Gaussian white noise of variance 1, $u(k) \sim N(0,1)$, was used to excite the system. Figure 4 shows the first 100 input and output data records. The input pattern presented to the binary neural net was given as $X(k) = [1 \ x(k-1) \ u(k-1)]^T$ and the adaptation step μ was set to 0.1. Equation (14) was chosen as the candidate tolerance function for each neuron with $\beta = 0.99$, $\Gamma = 5$ and $\alpha = 1$. With the specified parameters β, Γ and α, the tolerance function of eqn.(14) is a function which shrinks in time to satisfy the property of eqn.(8). Initially weights for all neurons were set to zero. Using the algorithm described in section 4 a two layer binary tree neural net was formed as shown in Fig.5. The neuron 00 was the initial neuron and neurons 10 and 11 were two related neurons produced and linked by the algorithm. The profiles of the tolerance functions for the three neurons are shown in Fig.6 and these are monotonically decreasing satisfying the requirement of eqn.(8). Figure 7 shows the misclassification rate or error for each presented pattern. As the number of presented patterns was large, the error converged to zero indicating that asymptotically the misclassification rate of eqn.(13) also converged to zero. An optimal classification system was therefore obtained. Profiles of the weight vectors for the three neurons are shown in Fig.8. The three weight vectors converged to a steady state. When 1000 patterns had been presented to the network, the weight vectors for the three neurons were given as

$$
\begin{aligned}
W_{00}(1000) &= [0.0205 \quad 0.5446 \quad 0.9001]^T \\
W_{10}(1000) &= [0.1027 \quad 0.3037 \quad 0.6964]^T \\
W_{11}(1000) &= [-.3339 \quad 0.5924 \quad 0.5959]^T
\end{aligned}
$$

where $W_{00}(.)$, $W_{10}(.)$ and $W_{11}(.)$ denote the weight vector for the neurons 00, 10 and 11 respectively. After 10000 patterns were fed to the network, the weights for the three neurons converged to

$$
\begin{aligned}
W_{00}(10000) &= [0.0002 \quad 0.4994 \quad 0.9992]^T \\
W_{10}(10000) &= [0.6000 \quad 0.5000 \quad 0.4000]^T \\
W_{11}(10000) &= [-.6000 \quad 0.5000 \quad 0.4000]^T
\end{aligned}
\tag{18}
$$

A comparison of the estimated weight vectors at time instant 10000 with the original system equations of eqn.(17) shows that the estimated vectors have converged to the true parameters. at the three operating points. The parameter estimates for neurons 10 and 11 are accurate and unbiased estimates, but the parameter estimates for neuron 00 which described pattern vectors belonged to Ω_0 ($X(K) \in \Omega_0$), are slightly in error because of the non-zero tolerance limit $\gamma_{00}(k)$ which has a value of 0.0529 at $k = 10000$. Hence data vector $X(k) \notin \Omega_0$ but with $|e(k)| < \gamma_{00}(k)$ would still be classified as belonged to Ω_0 and this introduces a small bias in the weight vector $W_{00}(k)$. This bias could be easily eliminated with the optimisation procedure described in section 6.

5.1.1 Discriminant functions

If the class membership of an input pattern vector could be pre-determined, without knowing the reference output of the neural net, an appropriate class of model for the system could be pre-selected and a prediction of the system output could then be obtained based on the selected model. This prediction capability can easily be achieved with a slight modification to the binary tree neural network. Consider the fitted model for $X(k) \in \Omega_0$, the modelling error is defined as

$$\epsilon(k) = y^*(k) - X(k)^T W_{00}(k) \quad , \quad |\epsilon(k)| \leq \gamma_{00}(k) \tag{19}$$

where $\gamma_{00}(.)$ denotes the tolerance function for neuron 00. Now if the pattern vector belongs to $X(k) \in \Omega_+$ and not Ω_0 the reference output $y^*(k)$ can be replaced by the corresponding estimate $X(k)^T W_{11}(k)$ to give

$$X(k)^T \left[W_{11}(k) - W_{00}(k) \right] = \epsilon(k) > \gamma_{00}(k) \quad \rightarrow \quad X(k) \in \Omega_+ \tag{20}$$

Similarly for input pattern vector $X(k) \in \Omega_-$,

$$X(k)^T \left[W_{10}(k) - W_{00}(k) \right] = \epsilon(k) < -\gamma_{00}(k) \quad \rightarrow \quad X(k) \in \Omega_- \tag{21}$$

From eqns.(20) and (21), the estimated weights for individual neurons can be bonded together to form discriminant functions or partition planes for data classification.

Hence at the time instance $k = 10000$ say the discriminant functions for system S_1 could be obtained by combining weight vector $W_{10}(10000)$ to $W_{00}(10000)$ and the tolerance limit $\gamma_{00}(10000) = 0.0529$ to form

$$0.5998 + 0.0006x(k-1) - 0.5992u(k-1) < -0.0529 \quad \rightarrow \quad X(k) \in \Omega_-$$

or

$$u(k-1) - 0.0010x(k-1) > 1.0893 \quad \rightarrow \quad X(k) \in \Omega_- \tag{22}$$

and $W_{11}(10000)$ to $W_{00}(10000)$ to give

$$u(k-1) - 0.0010x(k-1) < -1.0900 \quad \rightarrow \quad X(k) \in \Omega_+ \tag{23}$$

The discriminant functions of eqns.(22) and (23) are of the same form as the discriminant functions of the original system eqn.(17).

5.2 Stochastic multi-class systems

In the case of stochastic multi-class system identification the operation of the self-growing binary neural net is essentially the same except that the input pattern vector $X(k)$ has to be modified such that the effects of noise on the pattern classification is minimised. Consider the same multi-class system of eqn.(17) but with the output of the system corrupted by output additive noise $\zeta(k) \sim N(0,0.0001)$ which is uncorrelated with the system input. The corrupted output for the stochastic system (S_2) is defined as

$$z(k) = x(k) + \zeta(k) \tag{24}$$

To reduce the effects of noise on the estimated weight vectors the input pattern is modified to $X(k) = [1 \ \hat{z}(k-1) \ u(k-1)]^T$ where the output data records are replaced by the filtered output responses

$$\hat{z}(k) = X(k)^T W_i(k) \tag{25}$$

where $W_i(k)$ denotes the weight vector of the neuron which $X(k)$ is classified to. The adaptation step μ was set to 0.05 and eqn.(14) was selected as the candidate for the tolerance function with $\beta = 0.99$, $\Gamma = 5$ and $\alpha = 1$. Applying the self-growing binary tree neural network algorithm produced the same two layer binary tree with three neurons as shown in Fig.5. The profiles of the tolerance functions for the three neurons are shown in Fig.9. Fig.10 indicates that the misclassification error for each pattern presented converges to a threshold as the number of data patterns becomes large. Profiles of the weight vectors for the three neurons are shown in Fig.11. When 1000 patterns were presented to the network, the weight vectors for the three neurons became

$$\begin{aligned} W_{00}(1000) &= [0.0017 \quad 0.5017 \quad 0.9637]^T \\ W_{10}(1000) &= [0.3096 \quad 0.4770 \quad 0.5556]^T \\ W_{11}(1000) &= [-.1368 \quad 0.5386 \quad 0.6793]^T \end{aligned}$$

After 10000 patterns were presented, the weight vectors converged to

$$\begin{aligned} W_{00}(10000) &= [-.0021 \quad 0.4963 \quad 0.9996]^T \\ W_{10}(10000) &= [0.5987 \quad 0.5029 \quad 0.4050]^T \\ W_{11}(10000) &= [-.5995 \quad 0.4944 \quad 0.4028]^T \end{aligned} \tag{26}$$

Compared to the weight vectors at time instant 10000 obtained with the deterministic system equations of eqn.(17) the estimated vectors are very close to the true parameters of the system. This demonstrates the capability of the proposed algorithm for stochastic multi-class system identification with a high signal to noise ratio. Incorporating the tolerance limit at time instant 10000 for the neuron 00 which was given as $\gamma_{00}(10000) = 0.0530$ with the weight vectors $W_{00}(.)$, $W_{10}(.)$ and $W_{11}(.)$ produced the discriminant functions

$$u(k-1) - 0.0111\hat{z}(k-1) > 1.0996 \quad \rightarrow X(k) \in \Omega_-$$
$$u(k-1) + 0.0032\hat{z}(k-1) < -1.0898 \quad \rightarrow X(k) \in \Omega_+ \tag{27}$$

and the estimated discriminant functions are again of a similar form to the original deterministic system of eqn.(17). Notice that a small value of adaptation step μ was selected in this case so that asymptotically the contribution of the noise to the estimated weight vector was reduced. However a small μ slows down the learning rate of the algorithm.

When the power of the output noise was increased by 20dB such that $\zeta(k) \sim N(0,0.01)$, the self-growing binary neural net algorithm was re-applied with $\mu = 0.002$, $\beta = 0.99, \Gamma = 5$ and $\alpha = 2$ and a two layer binary net resulted as shown in Fig.5. An even smaller μ was selected in this case in order to asymptotically reduce the effects of the noise on the estimated weight vectors and a higher α was selected such that the asymptotic threshold of the tolerance limit was wider to cope with the larger noise variance. Referring to this system as S_3 the profile of the tolerance functions for the three neurons are shown in Fig.12. Fig.13 shows the misclassification error for each presented pattern. Profiles of the weight vectors for the three neurons are shown in Fig.14. When 30000 patterns were presented to the network, the weight vectors for the three neurons converged to

$$W_{00}(30000) = [-.0058 \quad 0.4997 \quad 0.8712]^T$$
$$W_{10}(30000) = [-.0606 \quad 0.4624 \quad 0.6561]^T \tag{28}$$
$$W_{11}(30000) = [0.1244 \quad 0.4865 \quad 0.6547]^T$$

with a tolerance limit for neuron 00 of $\gamma_{00}(30000) = 0.3318$. Although the weight vectors are heavily biased the discriminant functions for the network obtained by combining the weights for the three neurons with the tolerance limit $\gamma_{00}(20000) = 0.3318$

$$u(k-1) + 0.1734\hat{z}(k-1) > 1.2878 \quad \rightarrow X(k) \in \Omega_-$$
$$u(k-1) + 0.0610\hat{z}(k-1) > -.9312 \quad \rightarrow X(k) \in \Omega_+ \tag{29}$$

are of the same form as the discriminant functions of the deterministic system of eqn.(17). A model predicted output for the fitted network is shown in Fig.15 and a reasonable output prediction was produced.

The bias in the estimated vectors might be due to the poor definition of the partition plane in the presence of noise. Data patterns close to the edge of the partition plane might be wrongly classified in the presence of noise. This misclassification rate gets worst as the noise power is increased because more data patterns would be misclassified. The inclusion of the misclassified data patterns in the training of the net therefore produces biased results. In the high signal to noise ratio case, this effect is less severe and the original system structure can still be retained. Alternative forms of noise modelling may alleviate this difficulty and these are currently under investigation.

The three simulated examples show that the accuracy of the weight vectors is governed by the separability of the data patterns and the terminal tolerance limit. Because of the finite number of data patterns which are available in practice even if the patterns were completely separable the terminal tolerance limit $\gamma(k)$ will not converge to zero. Hence all data patterns which originally belonged to Ω_0 would be classified to either Ω_+ or Ω_-. This will induce bias in the estimation. This problem becomes worse when the system under investigation is stochastic because the partition planes between different classes of patterns can become poorly defined.

6. Optimisation procedure

One drawback of the self-growing binary tree neural net algorithm is that the estimated model will be biased towards the Ω_0 set if the number of data patterns for learning and adaptation is finite. The terminal threshold of the tolerance limit $\gamma_{00}(k)$ will not converge to zero even if the system is deterministic and the estimates obtained are no longer optimal. At regions close to the edges of the tolerance limit $\gamma_{00}(k)$ a smaller misclassification error may be obtained if the data pattern is classified to either Ω_+ or Ω_- instead. An optimisation procedure can therefore be derived based upon this feature.

One possible way to achieve the optimisation off-line is to carry out an exhaustive search on all patterns after the initial estimation in such a way that the final estimates produce the least misclassification rate. An off-line optimisation procedure to implement this can be summarised as follows:-

i) Assume that there are n classes of patterns and there are $N_1, N_2, \ldots, N_n$ data patterns classified to $\Omega_1, \Omega_2, \ldots, \Omega_n$ respectively. Record the misclassification error for each pattern.

ii) From all the available misclassification errors, select the pattern $X(k)$ which contributes the maximum cost to the misclassification rate, say $X(k) \in \Omega_1$, and pass this pattern $X(k)$ to the rest of the classes $\Omega_2, \Omega_3, \ldots, \Omega_n$ for classification. The one that contributes the least misclassification error, say Ω_2, is selected as the correct class for data pattern $X(k)$.

iii) Set $N_1 = N_1 - 1$, $N_2 = N_2 + 1$ and perform a new training for the data patterns belonging to Ω_1 and Ω_2. Record the misclassification errors for the new training process.

iv) Repeat procedures ii) and iii) until there is no further reduction in the misclassification rate. The final estimate should therefore become an optimal solution to the data set.

One drawback of the exhaustive search method is that it is a very time consuming process even if it is performed off-line. Implementing the optimisation procedure on-line is almost impossible. An alternative on-line optimisation procedure is therefore proposed. Consider the case when the tolerance limit of the self-growing binary tree neural net reaches a certain threshold then the partitioning and learning algorithm switches to the on-line optimisation procedure described below:-

i) When a new pattern $X(k)$ arrives it is passed to all the neurons within the binary tree and all misclassification errors for individual neurons are recorded.

$$\epsilon_i(k) = y^*(k) - X(k)^T W_i(k) \quad ; \quad i=1,\ldots,n \tag{30}$$

where $W_i(k)$ is the weight vector associated with the i'th neuron associated with data patterns belonging to Ω_i.

ii) The neuron which contributes the least absolute misclassification error is then chosen.

$$\epsilon(k) = \epsilon_j(k) = \min[|\epsilon_i(k)| \; ; \; i=1,\ldots,n] \;\rightarrow\; X(k) \in \Omega_j \tag{31}$$

iii) The weight vector for the j'th neuron is updated using the LMS algorithm

$$W_j(k+1) = W_j(k) + 2\mu\varepsilon(k)X(k) \tag{32}$$

Notice that eqns.(30), (31) and (32) are simple, efficient, can easily be implemented on-line and will work in association with the self-growing binary tree neural network algorithm.

6.1 Illustrative examples

For the deterministic system S_1 the weight vector of eqn.(18) was chosen as initial starting estimates and the on-line optimisation procedure of eqns.(30), (31) and (32) was applied to the newly arrived patterns. After a further 100 patterns were presented to the optimisation procedures, the weight vectors converged to

$$\begin{aligned} W_{00}(100) &= [0.0000 \quad 0.5000 \quad 1.0000]^T \\ W_{10}(100) &= [0.6000 \quad 0.5000 \quad 0.4000]^T \\ W_{11}(100) &= [-.6000 \quad 0.5000 \quad 0.4000]^T \end{aligned} \tag{33}$$

When the optimisation procedure was applied to the stochastic system S_2 and the estimated weight vectors of eqn.(26) were taken as the starting estimates the weight vectors converged to

$$\begin{aligned} W_{00}(5000) &= [0.0001 \quad 0.5001 \quad 0.9997]^T \\ W_{10}(5000) &= [0.5974 \quad 0.5006 \quad 0.4024]^T \\ W_{11}(5000) &= [-.5982 \quad 0.4976 \quad 0.4009]^T \end{aligned} \tag{34}$$

after 5000 additional patterns were fed to the network.

For the stochastic system S_3, the optimisation procedure produced the weight vectors

$$\begin{aligned} W_{00}(5000) &= [-.0064 \quad 0.5193 \quad 1.0005]^T \\ W_{10}(5000) &= [-.1012 \quad 0.4731 \quad 0.6950]^T \\ W_{11}(5000) &= [0.1685 \quad 0.4873 \quad 0.6799]^T \end{aligned} \tag{35}$$

after 5000 additional patterns were fed to the network with the weight vectors of eqn.(28) as starting values.

For deterministic and stochastic systems S_1 and S_2, eqns.(33) and (34) illustrate that better estimates were obtained compared with the estimates of eqns.(18) and (26). For the stochastic system S_3, a reasonable estimate has been obtained for patterns belonging to Ω_0 because the weight vector $W_{00}(.)$ was closer to the true system parameters.

7. Conclusions

A self growing binary tree neural network has been developed for the identification of multi-class systems. For well defined and separable processes the proposed network can correctly classify the patterns into the respective classes and produce weight vectors which accurately describe the behaviour of the system at different operating points. Because of the simplicity and efficiency of the network it can easily be implemented as an on-line algorithm for the identification of multi-class systems. An optimisation procedure based upon the minimisation of the misclassification errors has also been developed and shown to produce improved estimates.

References

1. K.M. Tsang and S.A. Billings [1991]: "Linear and nonlinear multi-class system identification", submitted for publication.
2. K.M. Tsang [1992]: " Parametric identification of hysteretic systems", Proc. Singapore Intl. Conf. Intelligent Control and Instrumentation, vol. 1, pp 465-470.
3. F. Rosenblatt [1962]: Principles of neurodynamics - perceptrons and the theory of the brain, Washington D.C.: Spartan.
4. B. Widrow and M.E. Hoff [1960]: "Adaptive switching circuit", 1960 IRE WESCON Conv. Record, Part 4, pp 96-104.
5. M.L. Minsky and S. Papert [1969]: Perceptrons - An introduction to computational geometry, Cambridge, MA: MIT Press.
6. R.P. Lippmann [1987]: " An introduction to computing with neural nets", IEEE ASSP Magazine, April, pp 4-22.
7. B. Widrow and M.A. Lehr [1990]: "30 years of adaptive neural networks: perceptron, madaline, and backpropagation", Proc. IEEE, vol. 78, pp 1415-1442.
8. B. Widrow and S.D. Stearns [1985]: Adaptive Signal Processing, Englewood Cliffs, NJ: Prentice-Hall.

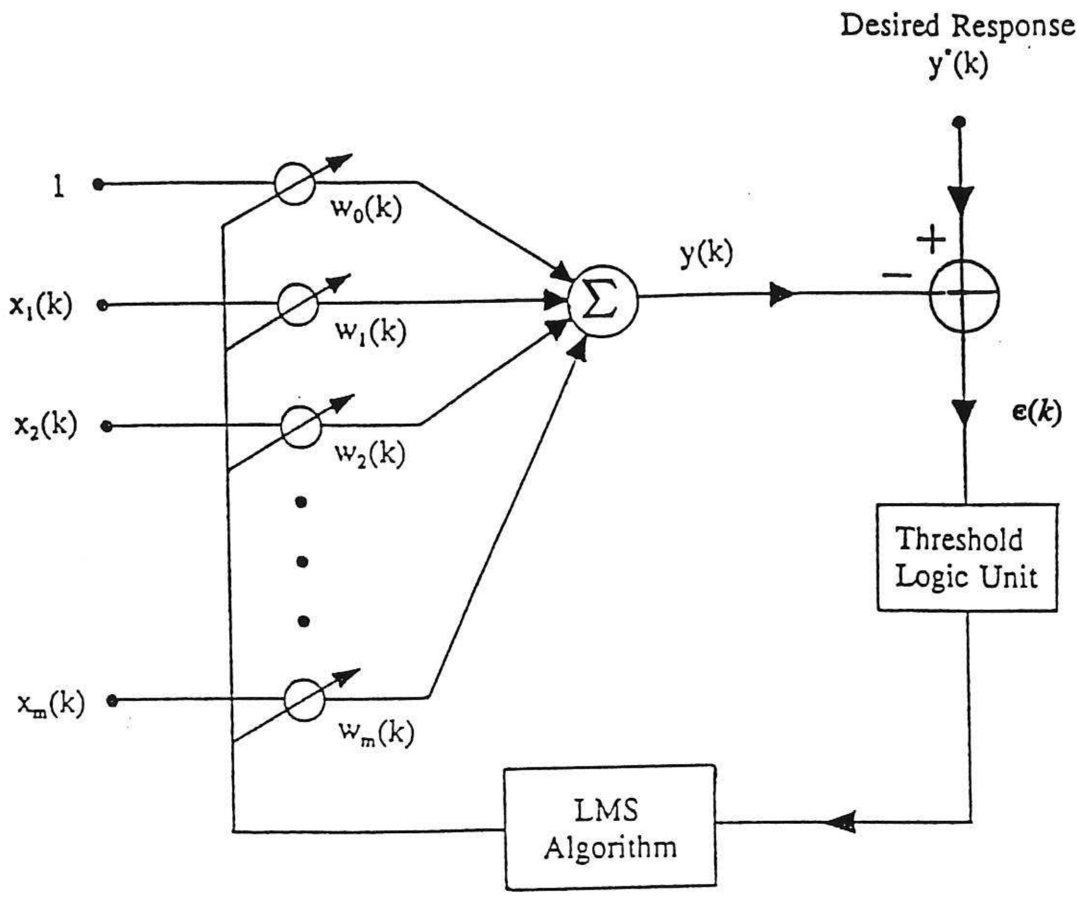
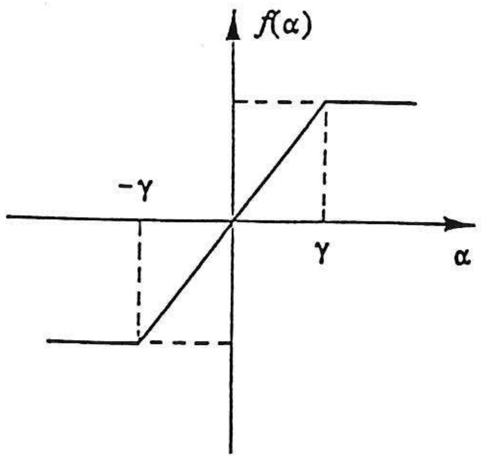


Figure 1. Single adaptive linear neuron



$$f(\alpha) = \alpha \quad , \quad |\alpha| \leq \gamma$$
$$f(\alpha) = \gamma \quad , \quad \alpha > \gamma$$
$$f(\alpha) = -\gamma \quad , \quad \alpha < -\gamma$$

Figure 2. Threshold logic unit

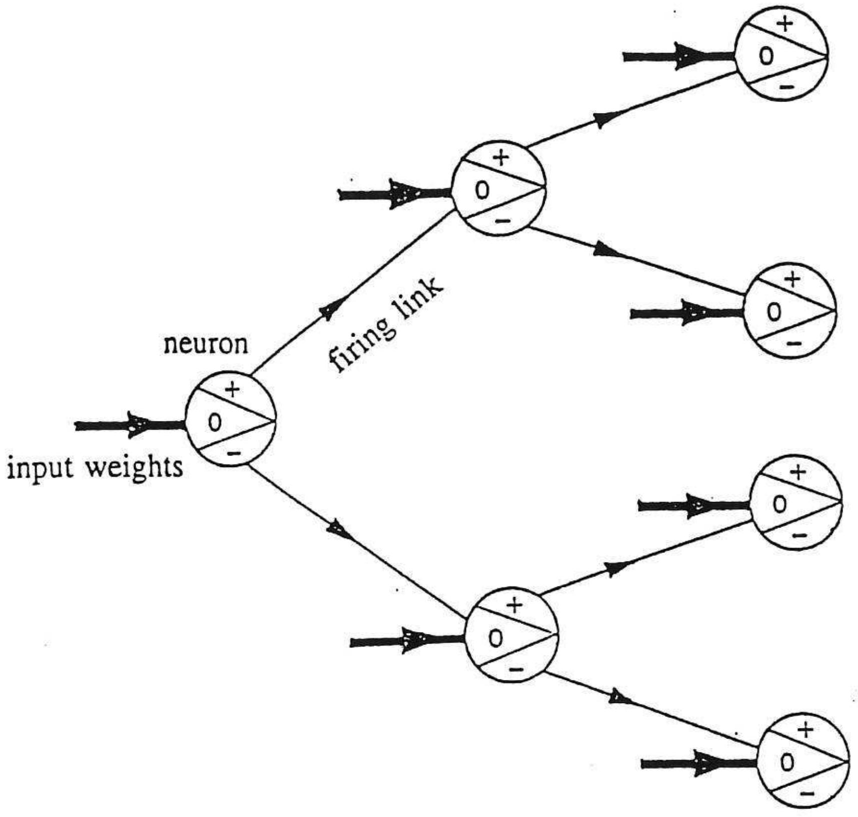


Figure 3. A three layer binary tree neural net

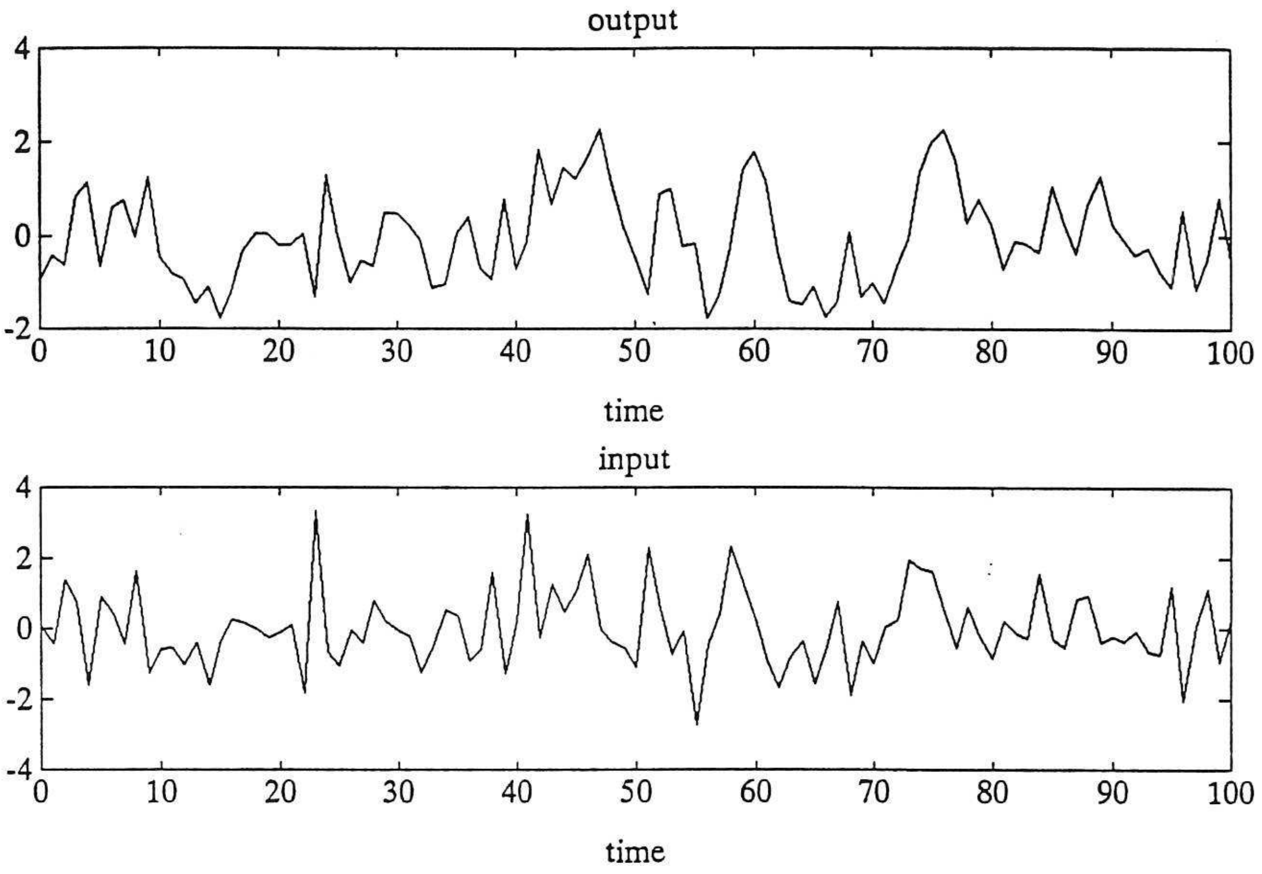


Figure 4. Input and output data records for system S_1

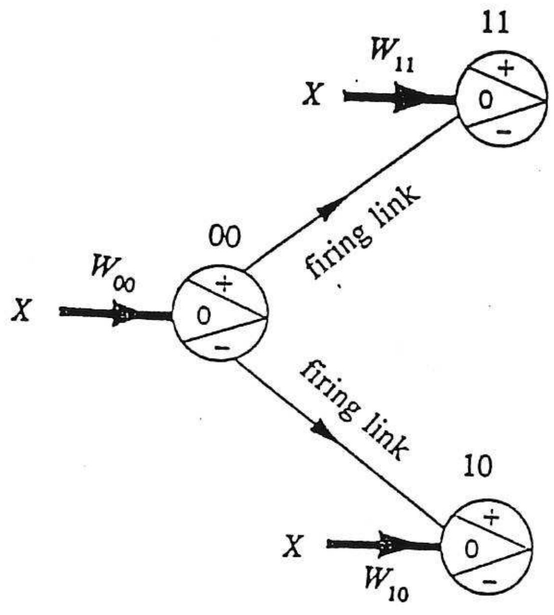


Figure 5. A two layer binary tree neural net for system S_1

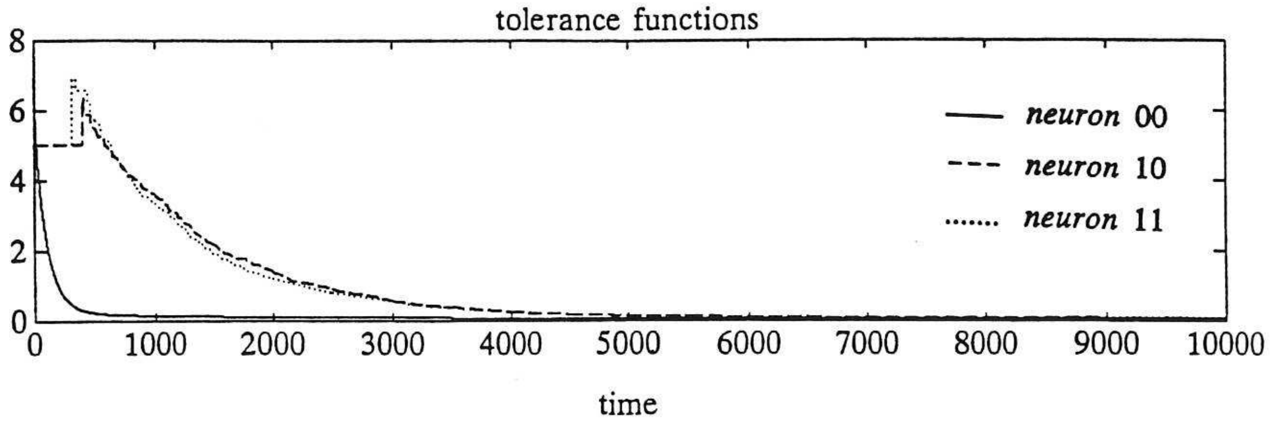


Figure 6. Profiles of the tolerance functions for system S_1

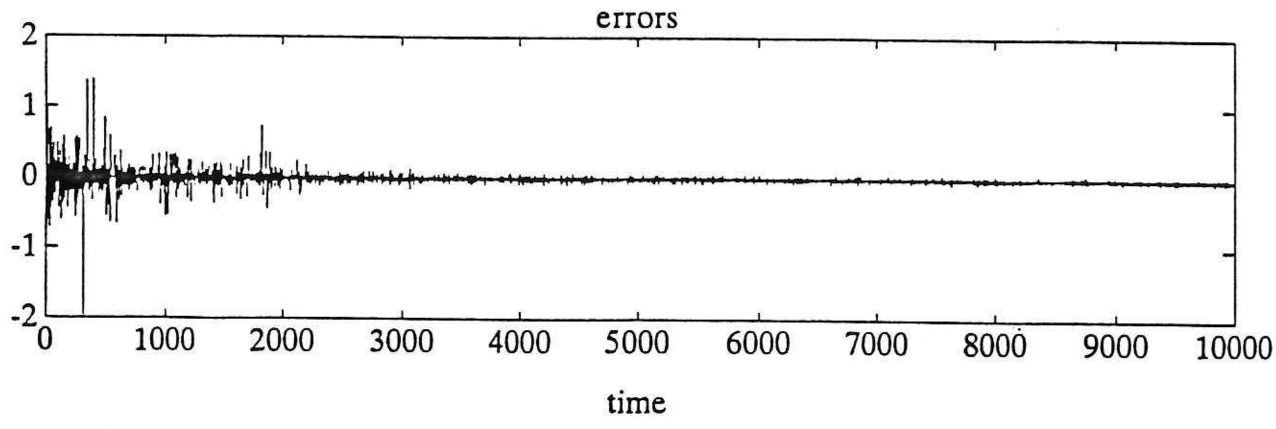
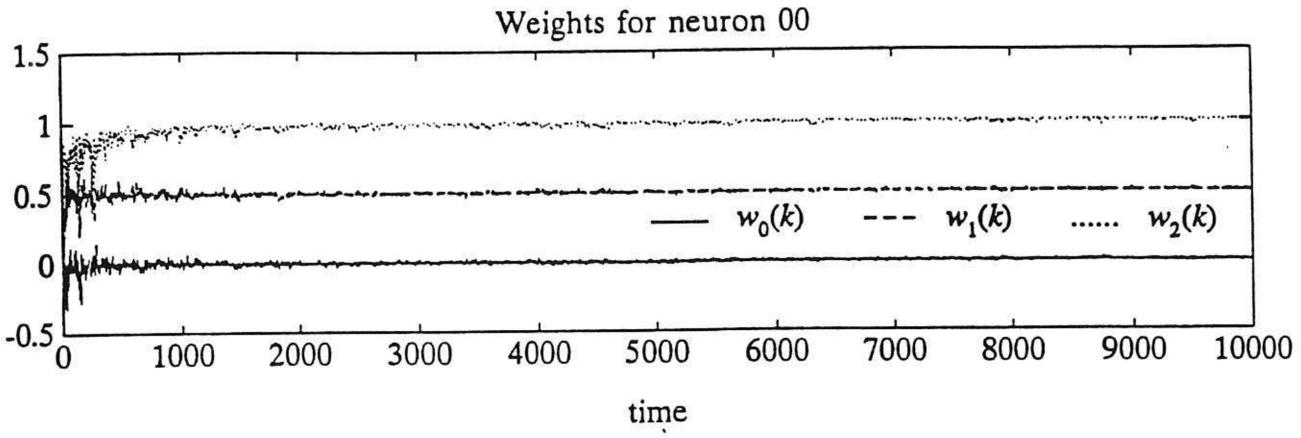
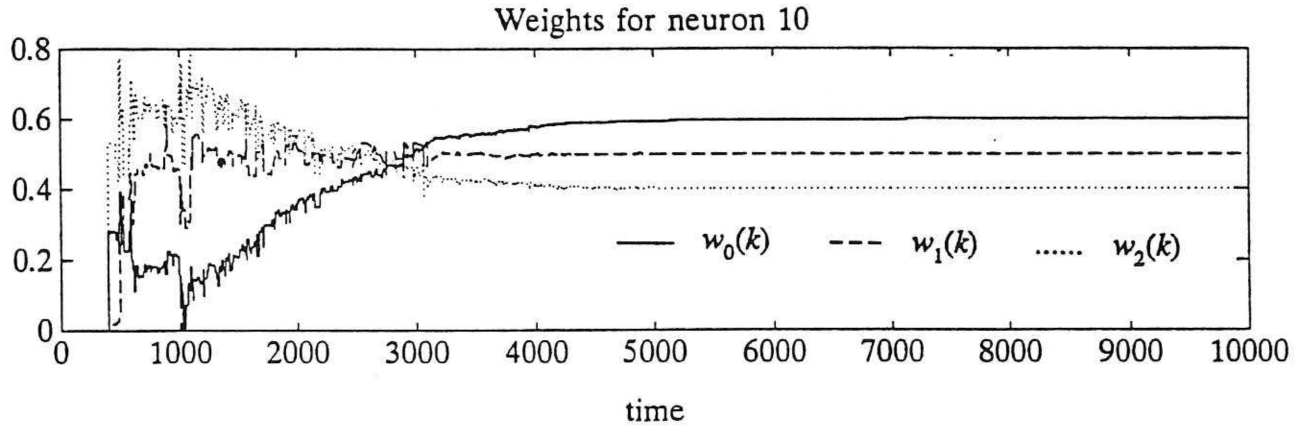


Figure 7. Profile of the misclassification rate for system S_1





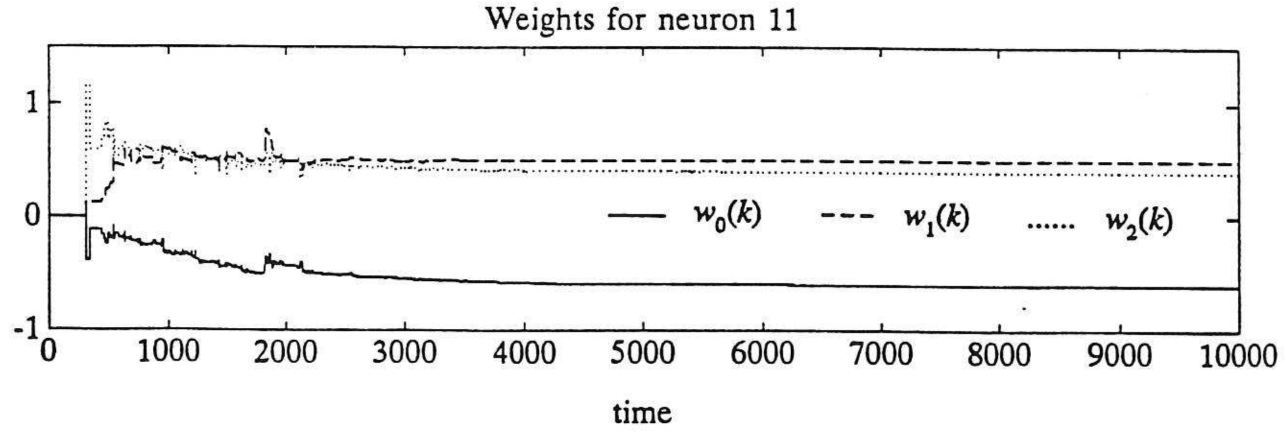


Figure 8. Profiles of the weight vectors for system S_1

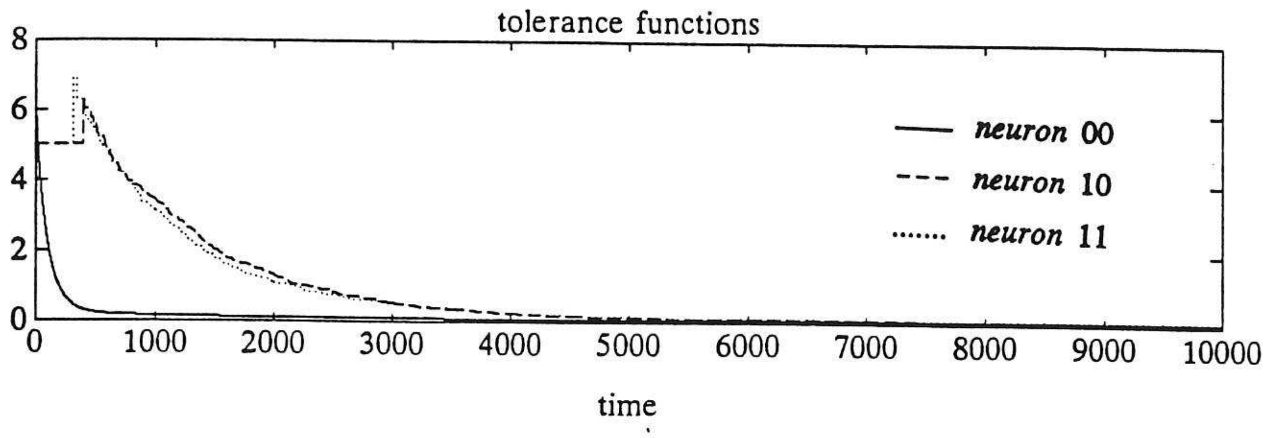


Figure 9. Profiles of the tolerance functions for system S_2

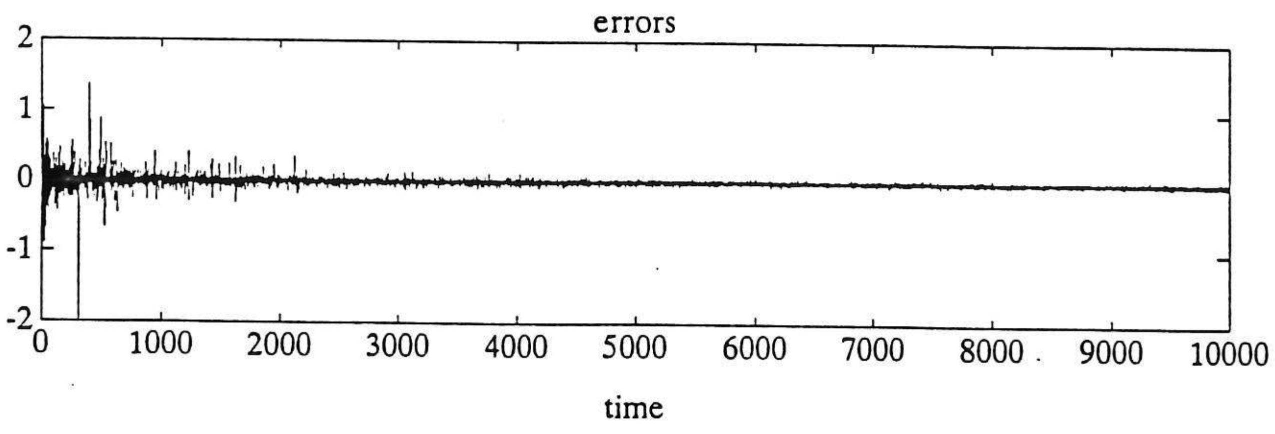


Figure 10. Profile of the misclassification rate for system S_2

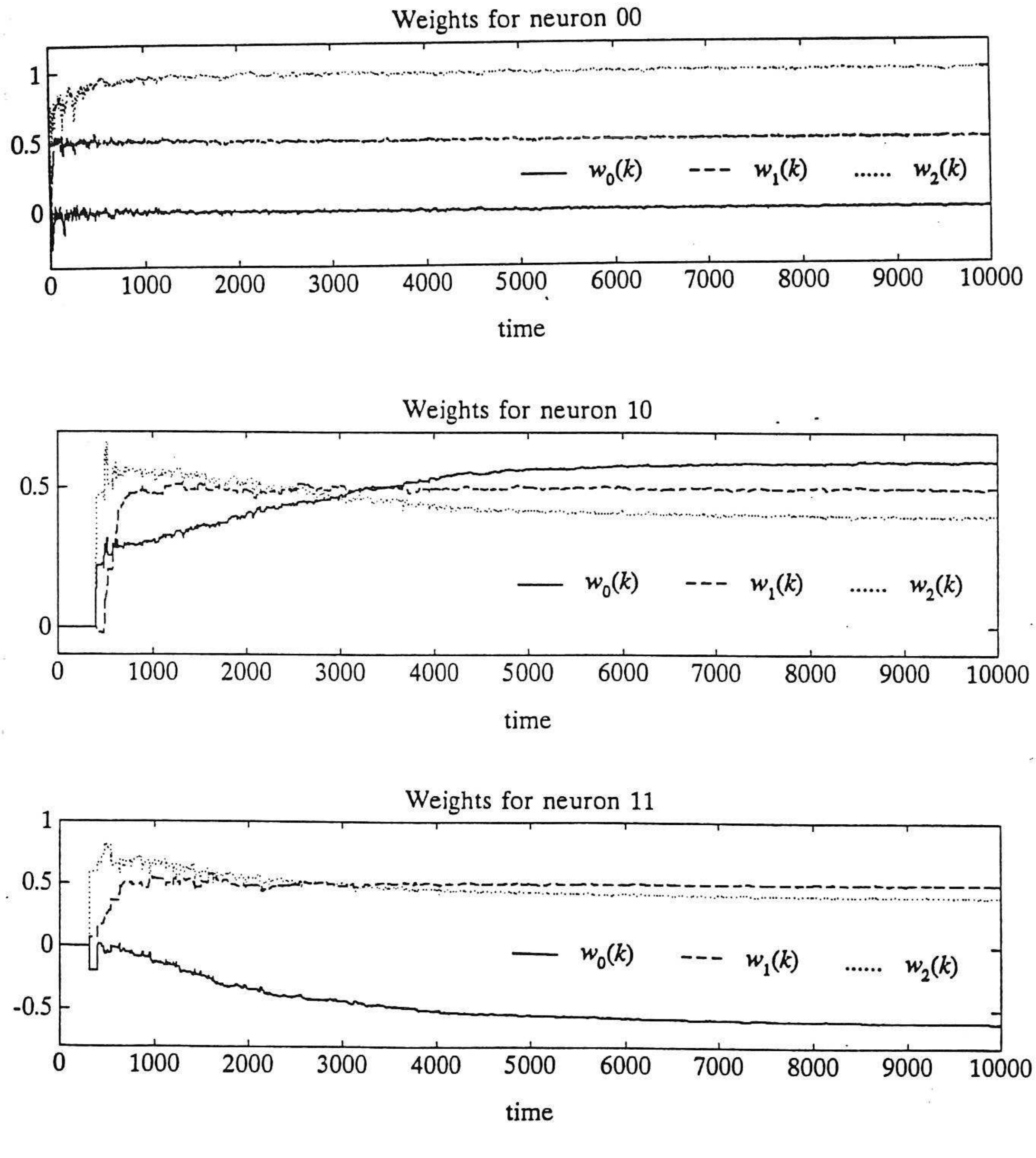


Figure 11. Profiles of the weight vectors for system S_2

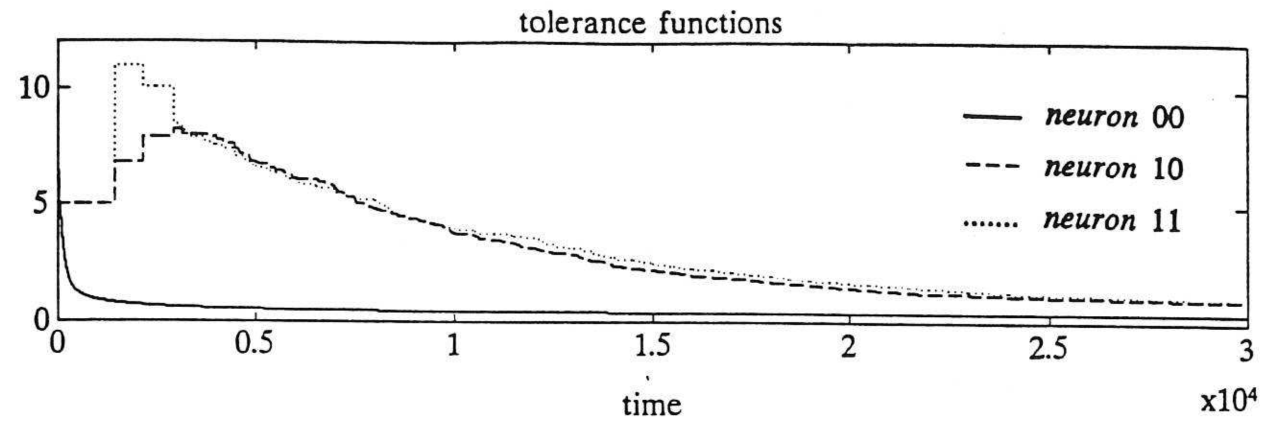


Figure 12. Profiles of the tolerance functions for system S_3

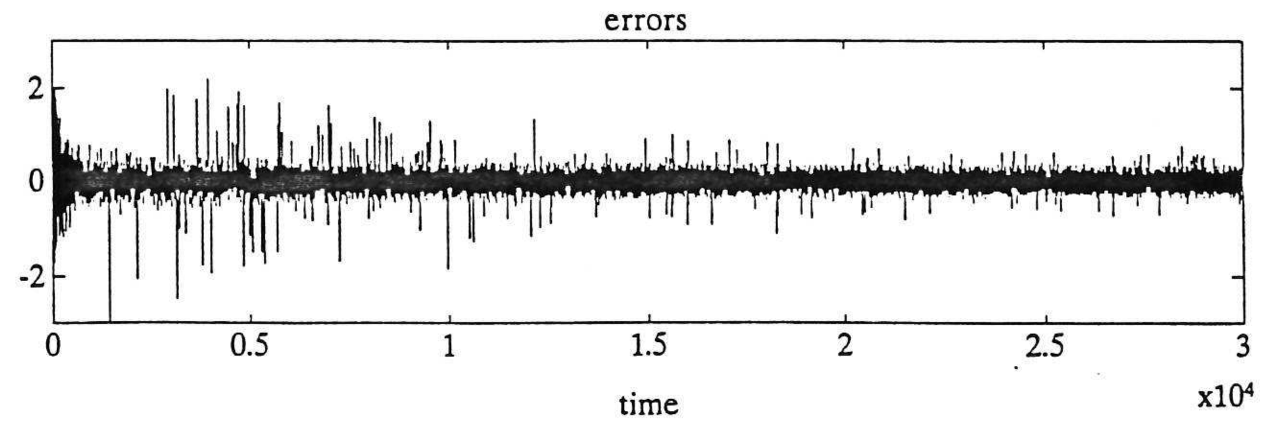


Figure 13. Profile of the misclassification rate for system S_3

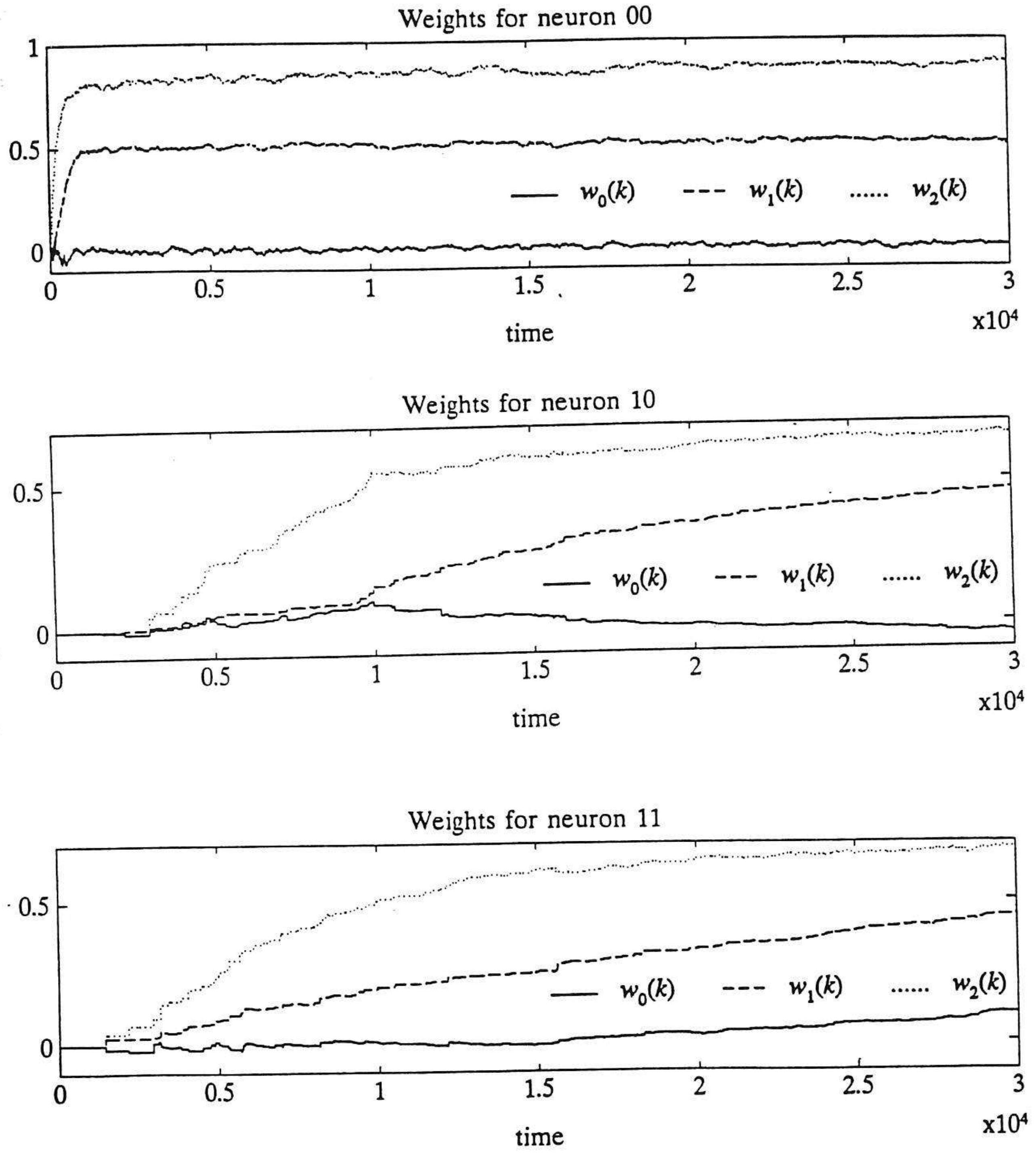


Figure 14. Profiles of the weight vectors for system S_3