CHAPTER V

ARTIFICIAL NEURAL NETWORK AND TESTING OF HYPOTHESIS

5.1 INTRODUCTION

It is a known fact that the hypothesis testing problem covers a very broad area of applications such as detection, classification and pattern recognition. Due to the significance of these types of applications, there has been much interest in developing neural network structure for hypotheses testing. In this Chapter, we will discuss the use of ANNs for solving hypothesis testing problems and the discussion is based on recent article by Pados and Papantoni-Kazakos (1995). Before we proceed, for the sake of completeness we give some notations and terminology regarding testing of hypotheses problem.

The problem of testing of hypotheses may be described as follows :

Let $\underline{x} = (x_1, x_2, \ldots, x_n)'$ be a random sample from a pdf $g_\theta(.)$ belonging to a family of distributions $\left\{ g_\theta(.),\ \theta \in \Theta \subset R^k \right\}$.

Let $\Theta = \{\theta_0, \theta_1\}$ and consider two hypotheses

$$H_0 : \theta = \theta_0 \qquad \text{versus} \qquad H_1 : \theta = \theta_1$$

For testing H_0 against H_1, let $\phi(\underline{x})$ be a test function of the form

$$\phi(x) = \begin{cases} 1 , & \text{if } \underline{x} \in R \\ 0 , & \text{if } \underline{x} \in A \end{cases}$$

where R and A denote the critical region and acceptance region respectively. Obviously, two types of errors are associated with such a test which are as follows :

1. $\alpha = P_\theta[$ type I error $] = P_\theta[$ reject $H_0], \text{ for all } \theta \in \Theta_0$

2. $\beta_\phi(\theta) = P_\theta[$ type II error $] = P_\theta[$ accept $H_0],$
$\text{for all } \theta \in \Theta_1$

Further, the power function is defined as

$$E_\theta\phi(x) = P_\theta[\ \underline{x} \in R\ ], \text{ for all } \theta \in \Theta$$

Now, for hypotheses testing problem where H_0 and H_1 are both simple, let $\left\{ g_\theta(.), \theta \in \Theta \right\}$, where $\Theta = \{\theta_0, \theta_1\}$, be a family of possible distributions of $\underline{x}$, and let us write $g_0(\underline{x})=g_{\theta_0}(\underline{x})$ and $g_1(\underline{x})=g_{\theta_2}(\underline{x})$ for convenience. Then Neyman-Pearson lemma (Lehmann, 1986) offers a Most Powerful(MP) test $\phi(\underline{x})$ for a given α which is of the form

$$\phi(\underline{x}) = \begin{cases} 1 & , \text{ if } g_1(\underline{x}) > k\, g_0(\underline{x}) \\ \nu(\underline{x}) & , \text{ if } g_1(\underline{x}) = k\, g_0(\underline{x}) \\ 0 & , \text{ if } g_1(\underline{x}) < k\, g_0(\underline{x}) \end{cases}$$

for some $k \geq 0$ and $0 \leq \nu(\underline{x}) \leq 1$ and $E_{H_0}(\phi(\underline{x})) \leq \alpha$.

As is well-known, the theory of testing of hypothesis is well developed and it is excellently documented in Lehmann (1986). So, we do not discuss further this topic and proceed to present an application of ANN modelling in testing of hypothesis problems.

We note that in the above approach, one needs the knowledge of distributional form(function) of $\underline{x}$ under H_0 and H_1. Clearly, in the absence of such knowledge, one cannot implement the above test procedure. Of course many other alternative procedures (such as Non-parametric procedures) exist. Recently, Pados and Papantoni-Kazakos (1995) made an attempt to offer neural network solution for testing of hypothesis problem.

In this Chapter, we focus our attention to use of ANN for testing simple versus simple hypothesis. In Section 2, we describe the ANN model proposed by Pados and Papantoni-Kazakos (P-PK) (1995) for this purpose. In Section 3, P-PK learning rule is discussed. At the end, an example is given to illustrate the theory followed by concluding remarks.

5.2 ANN MODEL FOR HYPOTHESES TESTING

Pados and Papantoni-Kazakos (1995) designed a different kind of ANN model shown in Fig.5.1 for testing a simple versus simple hypotheses.

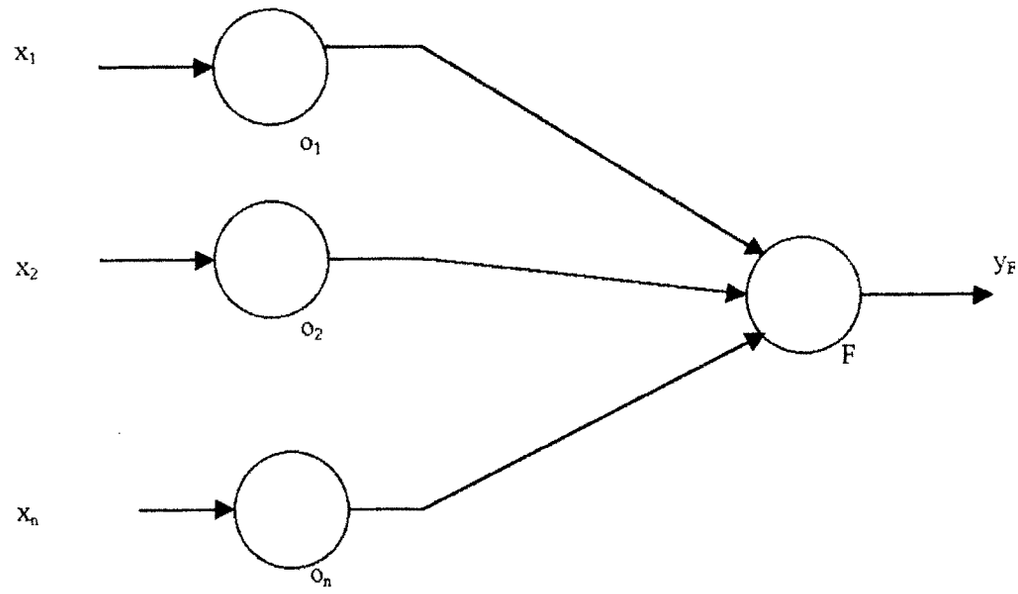

Figure 5.1 ANN Model for Hypothesis Testing problem

In the above model, $\underline{x} = (x_1, x_2, \ldots, x_n)'$ denotes the input vector, $\underline{o} = (o_1, o_2, \ldots, o_n)'$ denotes the output vector in the hidden layer, and final output of network is denoted by y_F.

Here, we note that structure of the ANN model as shown in Fig.5.1 differs(i.e. here x_i's are connected only to o_i's) from the earlier models which are discussed in Chapter II and III. However, the rule associated with this model is similar to that

given in (2.2.2). Here, the rule for computing the o_i's is as follows :

$$o_i = \begin{cases} 1 , & \text{if } f(x_i) > T_i \\ 0 , & \text{if } f(x_i) \leq T_i \end{cases} \qquad (5.2.1)$$

where $f(x_i)$ is an activation function (performed on input x_i) and T_i is its threshold value. Regarding the output of neuron F, its input consists of all the outputs produced by the previous neurons o_i's (i = 1,2,...,n), and let y_F be the output(which is the final output of ANN) computed as follows :

$$y_F = \begin{cases} 1 , & \text{if } f_F(\underline{o}) > T_F \\ 0 , & \text{if } f_F(\underline{o}) \leq T_F \end{cases} \qquad (5.2.2)$$

where $f_F(.)$ is an activation function (may be different from $f(.)$) and T_F is the corresponding threshold value. Since the objective of this network is hypotheses testing, following convention will be used for finding final output of network :

$y_F = 1$, when the network decides that its input $\underline{x}$ comes from hypothesis H_1.

$y_F = 0$, when the network decides that its input $\underline{x}$ comes from hypothesis H_0

The crucial part of the theory developed by the Pados and Papantoni-Kazakos (1995) is about choosing the activation function f(.) and f_F (.) which are called as 'suitable testing functions' or 'suitable functions'. Below we discuss the same.

5.2.1 Suitable Testing Functions

Definition 5.1 :

Let H_0 and H_1 be any two hypotheses. Then, a function f(x) is called 'suitable' for the specific hypothesis testing problem, if

$$P_1(f(x) > T) > P_0(f(x) > T),$$

$$\text{for any } T \in \left(\inf_x f(x) , \sup_x f(x) \right) \tag{5.2.3}$$

To illustrate the above definition, consider the following example :

Example 5.1 : Consider a class of hypotheses testing problem for testing H_0 versus H_1 as

$$H_0 : X \sim g_0(x) \qquad \text{VS} \qquad H_1 : X \sim g_0(x-\theta)$$

$$\text{for some } \theta > 0$$

Let

$$f(x) = x$$

Then, (5.2.1) become,

$$o = \begin{cases} 1 , & \text{if } x > T \\ 0 , & \text{if } x \leq T \end{cases}$$

From the Definition (5.1), it can be seen that above test function f(x) for testing H_0 versus H_1 is suitable because

$$P_1(X > T) > P_0(X > T), \quad \text{for every real } T$$

This implies that, any strictly monotonically increasing function f(x) is suitable.

REMARK 5.1 : From the above Example, it can be observed that any strictly monotonically increasing function is suitable. Therefore, frequently used sigmoid function is used as a testing function.

Below, we prove the Proposition for obtaining testing function $f_F(.)$.

Proposition 5.1 : If f(x) is suitable (in the sense described in Definition (5.1)) then the test function for the neuron F is

$$f_F(\underline{o}) = \sum_{i=1}^{n} w_i o_i , \qquad w_i > 0 \quad \text{for } i = 1,2,\ldots,n$$

a suitable test function for any arbitrary hypotheses testing problem.

Proof : We have

$$o_i = \begin{cases} 1 , & \text{if } f(x_i) > T_i \\ 0 , & \text{if } f(x_i) \leq T_i \end{cases}$$

$$\text{for all } i = 1,2,\ldots,n \qquad (5.2.4)$$

Since $f(x_i)$ is suitable function, we have

$$P_1(f(x_i) > T_i) > P_0(f(x_i) > T_i) , \; i = 1,\ldots,n$$

This implies

$$P_1(o_i = 1) > P_0(o_i = 0), \qquad \text{for all } i = 1,\ldots,n$$

then for $w_i > 0$

$$P_1(w_i o_i > T_F) > P_0(w_i o_i > T_F) \quad \text{for all } i = 1,\ldots,n$$

This implies that

$$P_1(\sum_{i=1}^{n} w_i o_i > T_F) > P_0(\sum_{i=1}^{n} w_i o_i > T_F),$$

$$\text{for any } T_F \in \left(0, \sum_{i=1}^{n} w_i \right) \qquad (5.2.5)$$

Therefore

$$f_F(\underline{o}) = \sum_{i=1}^{n} w_i o_i , \qquad (5.2.6)$$

is suitable for any arbitrary hypotheses testing problem.

□

Once the suitable test functions f(.) and $f_F(.)$ are selected, the next task is to train the network using a learning rule. We note here that Pados and Papantoni-Kazakos(P-PK) (1995) have pointed out that the traditional back-propagation learning rule is not a good choice for this problem and hence they have proposed a new rule which uses Neyman-Pearson approach.

5.3 P-PK LEARNING RULE

Consider the model presented in Fig 5.1. Assume that training set consists of a set of observations from hypothesis H_1, denoted by vector $\underline{x}_j = (x_{j1}, x_{j2}, \ldots, x_{jn})'$ and set of patterns from hypothesis H_0, by $\underline{z}_j = (z_{j1}, z_{j2}, \ldots, z_{jn})'$, $j = 1,2,\ldots,P$. The rule associated with the hidden layer neurons is given by

$$o_i = \begin{cases} 1, & \text{if } f(x_i) > T_i \\ 0, & \text{if } f(x_i) \leq T_i \end{cases}, \qquad (5.3.1)$$

for $i = 1,2,\ldots,n$

where T_i's ($i = 1,2,\ldots,n$) are threshold parameters. Then, the decision rule at neuron F is proposed as follows :

$$y_F = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} w_i o_i > M \\ \nu(x), & \text{if } \sum_{i=1}^{n} w_i o_i = M \\ 0 & \text{if } \sum_{i=1}^{n} w_i o_i < M \end{cases} \qquad (5.3.2)$$

where M is discrete valued threshold parameter and $\nu \in (0,1)$. Then, using the known set of patterns $\underline{x}_j$'s from H_1 and $\underline{z}_j$'s from H_0, the problem is to obtain threshold and weighting parameters w_i's so that the induced probability of first kind of error becomes equal to α.

In the following, we discuss the learning procedure referred to as 'Layer-by-Layer Learning' which recursively adapt the network parameters $\{T_1, T_2, \ldots, T_n, w_1, w_2, \ldots, w_n, M, \nu\}$ so that the the induced probability of error will eventually become equal to value α. (As pointed out by P-PK (1995), it can be seen that this approach is similar to N-P approach.)

5.3.1 Layer-by-Layer Learning Procedure

In this learning procedure, assume that, a set of patterns $\underline{z}_j$'s (j=1,2,...,P) from hypothesis H_0 is available and level of significance or a false alarm probability α is specified .

The first step in this procedure is to train the hidden

layer neurons to induce a false alarm probability equal to α. This is done by feeding a single hidden neuron i ($i \in \{1,2,\ldots,n\}$) with all Pn available observations and then use the 'best' estimate of this threshold T_i, say T^*, as the threshold for all hidden layer neurons.

To obtain such a T^*, a recursive formula is derived and it is as follows :

$$T_{k+1} = T_k + c_k \left(o(u_k, T_k) - \alpha \right) \quad \text{for } k = 1,2,\ldots,Pn \tag{5.3.3}$$

where u_k's are observations after relabelling the observations z_{ji}'s ($i=1,\ldots,n$; $j=1,\ldots,P$), T_1 is an arbitrary initial value, $c_k = c/k$ for some $c>0$ (where c is learning rate), and $o(.)$ is as given in (5.3.1).

NOTE : Note that, the value of T_i converges to the desired value with probability one and this is proved by P-PK (1995, pp 599)

Using the value of T^*, the threshold parameters M, ν given in (5.3.2) are evaluated so that the probability of first kind of error induced by the output neuron F equals α. For this, the corresponding recursive formulas are obtained which are as follows:

$$\nu_{j+1} = \nu_j - c_j\left(Y_F(\{ z_{(j-1)n+1}, z_{(j-1)n+2}, \ldots, z_{jn}\}, M_j, \nu_j) - \alpha \right)$$

$$M_{j+1} = M_j - [\nu_{j+1}]$$

$$\nu_{j+1} = \nu_{j+1} - [\nu_{j+1}], \qquad \text{for } j = 1,2,\ldots,P.$$

(5.3.4)

where [x] is greatest integer which is less than or equal to x, α is the pre-specified error probability, ν_1 is an arbitrary initial value in (0,1), M_1 is an arbitrary initial value in {0,1, ...,n}, and $c_j = c/j$, for some c>0 and j = 1,2,...,P.

NOTE : However, the expression (5.3.2) contains the weighting parameters w_i's, an optimal selection for the weighting parameters is suggested by P-PK (1995) as

$$w_i = 1 \quad \text{for} \quad i=1,2,\ldots,n.$$

Now, we illustrate how the ANN model given in Fig. 5.1 can be used to test a simple versus simple hypothesis testing problem using a simulated data.

Illustration

Let $\underline{x}$ be a random variable from a $N(\mu,\sigma^2)$, with $\sigma^2=1$. Consider the problem of testing

$$H_o : \mu = 0 \qquad \text{VS} \qquad H_1 : \mu = 1$$

Now, 300 patterns each consisting of 8 observations from N(0,1) and N(1,1) are generated and the maximum allowable false alarm rate is fixed at α=0.1

For testing H_o versus H_1, the learning algorithm discussed above is applied to the basic neural network structure shown in Fig. 5.1. For this, the number of hidden layer neurons is n=8 and testing function applied on these neurons is given by

$$f(x) = \frac{1}{1 + \exp(-x)}$$

As mentioned earlier, the testing function of the neuron F is as follows :

$$f_F(\underline{o}) = \sum_{i=1}^{n} w_i o_i , \qquad w_i > 0 \quad \text{for } i = 1,2,\ldots,n$$

Using the above generated data the network is trained.

After the training, test observations under both H_o and H_1 are applied to the network and the error rate observed is 0.095, that is, out of 100 cases, approximately 1 is rejected.

Using the same data, N-P test statistic was also computed and it was observed that the conclusion drawn by N-P method and P-PK method are similar.

Before we conclude, we note that the P-PK learning algorithm (for that matter many existing neural network learning algorithms including the famous back-propagation algorithm) is an application of Robbins and Monro stochastic approximation theorem (1951). Therefore, as pointed out by P-PK (1995), further learning algorithmic improvements are possible, for the literature on stochastic approximation is encouraging.

■