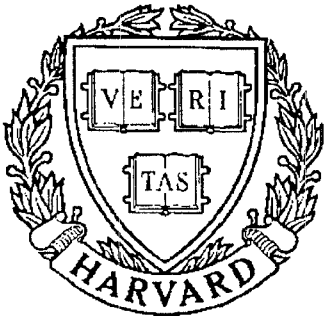
TECHNICAL RESEARCH REPORT

SYSTEMS RESEARCH CENTER



Supported by the National Science Foundation Engineering Research Center Program (NSFD CD 8803012), the University of Maryland, Harvard University, and Industry

Research support for this report has been partially provided by the Naval Research Laboratory/Systems Research Center Fellowship Program at the University of Maryland through ONR Grant N00014-85-G-0213

Neural Networks for Sequential Discrimination of Radar Targets

by J.A. Haimerl and E. Geraniotis

TR 91-26

NEURAL NETWORKS FOR SEQUENTIAL DISCRIMINATION OF RADAR TARGETS

J. A. Haimerl and E. Geraniotis

NEURAL NETWORKS FOR SEQUENTIAL DISCRIMINATION OF RADAR TARGETS

J. A. Haimerl and E. Geraniotis

Abstract

In this paper, perceptron neural networks are applied to the problem of discriminating between two classes of radar returns. The perceptron neural networks are used as nonlinearities in two threshold sequential discriminators which act upon samples of the radar return. The test statistic compared to the thresholds is of the form $T_n(\mathbf{Z}) = \sum_{j=1}^{n-K+1} \gamma(Z_j, Z_{j+1}, \ldots, Z_{j+K-1})$ where Z_i, $i = 1, 2, 3, \ldots$ are the radar samples and $\gamma()$ is the nonlinearity formed by the neural network. Numerical results are presented and compared to existing discrimination schemes.

Dr. Geraniotis is with the Department of Electrical Engineering and the Systems Research Center of the University of Maryland, College Park, MD 20742.

Joseph A. Haimerl was a Naval Research Laboratory/Systems Research Center Fellow at the University of Maryland; he is now with GE Aerospace, Government Electronic Systems Division, Moorestown, NJ 08057.

This research was supported by the Naval Research Laboratory/Systems Research Center Fellowship Program at the University of Maryland through ONR Grant N00014-85-G-0213.

NEURAL NETWORKS FOR SEQUENTIAL DISCRIMINATION OF RADAR TARGETS

I. Introduction and Motivation

A problem often encountered in military radar applications is the binary classification problem. That is, after detecting an object, the radar must classify it as a member of one of two classes: target or decoy. We shall denote these two classes (or hypotheses) as H_1 and H_0 respectively.

The discriminator (*i.e* classification or automatic target recognition circuitry) obtains a sequence of data, $Z_1, Z_2, \ldots, Z_n$, denoted by the vector $\mathbf{Z}$, from the radar returns of the detected object. This sequence may be samples of the radar return envelope signal, the in-phase and quadrature signals, the phase signal, or any other information carrying signal. In any case, the data sequence is a random process and is described by the probability density functions (pdfs) $f_i(z_1, z_2, \ldots, z_n)$ where $i = 0, 1$ represents hypothesis H_i. More specifically, we consider

$$\begin{aligned} H_1 &: \{Z\}_{i=1}^n \text{ has pdf } f_1(z_1, z_2, \ldots, z_n) = f_1(\mathbf{z}) \\ H_0 &: \{Z\}_{i=1}^n \text{ has pdf } f_0(z_1, z_2, \ldots, z_n) = f_0(\mathbf{z}) \end{aligned} \tag{1}$$

where $\mathbf{z}$ represents the n-tuple $(z_1, z_2, \ldots, z_n)$. It is assumed that the data sequence is stationary, and that the data sequence may be correlated and non-Gaussian.

For the discriminator to work properly, it must be designed so that the probability of it making an error is small. Quick decisions are also desirable, so n should be relatively small. If the probability density functions of the data were known, likelihood ratio functions could be used for the test. A likelihood ratio test has the form

$$d(\mathbf{Z}) = \begin{cases} 1, & \text{if } \frac{f_1(\mathbf{Z})}{f_0(\mathbf{Z})} \geq \eta; \\ 0, & \text{if } \frac{f_1(\mathbf{Z})}{f_0(\mathbf{Z})} < \eta. \end{cases} \tag{2}$$

where η is a constant to be determined. Hypothesis H_i is chosen by the discriminator when $d(\mathbf{Z}) = i$, $i = 0, 1$. Likelihood ratio tests are well known and optimal in the Bayes, Neyman-Pearson, and minimax senses[1]; the choice of η depends upon which criterion the designer chooses to optimize.

Typically, the n-dimensional probability density functions are not known or result in a structure too complex to implement. If the lower order probability density functions (*i.e.* with order $< n$) are known or can be estimated, discriminators with a test statistic of the form

$$T_n(\mathbf{Z}) = \sum_{j=1}^{n-K+1} g(Z_j, Z_{j+1}, \ldots, Z_{j+K-1}) \tag{3}$$

can be developed. The function $g(x_1, x_2, \ldots, x_K)$ is a suitably chosen nonlinearity. The test statistic can be used in both block and sequential tests. The fact that $K < n$ implies that a discriminator will have a simpler structure since it need only have a memory of K samples as opposed to n. The nonlinearity $g(x_1, x_2, \ldots, x_K)$, a function of only K variables, will be simpler in general than a function of n variables when $K < n$. However, since $K < n$, the test in general will be suboptimal to the n-dimensional likelihood ratio test.

A block test could be implemented as

$$d(\mathbf{Z}) = \begin{cases} 1, & \text{if } T_n(\mathbf{Z}) \geq \eta; \\ 0, & \text{if } T_n(\mathbf{Z}) < \eta. \end{cases} \tag{4}$$

where η is a decision threshold.

A sequential test could proceed as follows: compare the test statistic T_n to two thresholds a and b. If $T_n \geq b$ declare H_1 and terminate the test. If $T_n \leq a$ declare H_0 and terminate the test. If $a < T_n < b$ then obtain the next sample Z_{n+1}, compute T_{n+1}, and repeat the test. Terminate the sequential test with a block test if a decision has not been made by the maximum number of data samples N.

Memoryless tests (*i.e.* $K = 1$) have been studied by [2] and [3] for block and sequential tests respectively. [2] and [3] employ central limit theorems for dependent observations (especially for stationary mixing processes) to derive linear integral equations which they solve for the optimal nonlinearity $g(x)$. More recent work has been done for the one-step-memory case where $K = 2$ (see [4]). [4] showed that probability density functions up to degree 4 were needed to solve for the one-step-memory nonlinearity.

Estimation of probability density functions above degree 2 becomes impractical due to computer processing limitations, and knowledge of higher order pdfs is not likely. Therefore, a different approach to designing a discriminator is presented in this paper. Using apriori data sequences from each hypothesis (*i.e.* field data), a perceptron neural network is trained with a supervised learning algorithm to produce desirable outputs. This perceptron neural network is then implemented in a structure to act as the nonlinearity $g(x_1, x_2, \ldots, x_K)$ in equation (3). First, we shall review some basics of perceptron neural networks.

II. Perceptron Neural Network Basics

For a thorough introduction to perceptron neural networks, refer to [5]. Perceptron neural networks are interconnected layers of simple processing units called perceptrons. A perceptron is specified by its weighting vector $\mathbf{w} = (w_0, w_1, \ldots, w_{K-1})^T$, its offset value θ, and its nonlinearity. Usually the nonlinearity is a sigmoid

$$f(y) = \frac{1}{1 + e^{-y}}. \tag{5}$$

Note that we use the terms perceptron and node interchangeably. We also refer to the offset value θ as the node offset value.

The perceptron operates by taking the input vector
$\mathbf{x} = (x_0, x_1, \ldots, x_{K-1})^T$ and forming the dot product

$$\sum_{i=0}^{K-1} x_i w_i = \mathbf{x}\mathbf{w}^T. \tag{6}$$

From the dot product, an offset value θ is subtracted to get the result $y = \mathbf{x}\mathbf{w}^T - \theta$; y is then passed through the nonlinearity, and $f(y)$ is output. The nonlinearity $f(y)$ limits the outputs to values between 0 and 1. Very large values of y will result in values of $f(y)$ near 1. As y approaches $-\infty$, $f(y)$ approaches 0.

If the sigmoid were replaced by a hard limiter,

$$q(x) = \begin{cases} 1, & \text{if } x \geq 0; \\ 0, & \text{otherwise,} \end{cases} \tag{7}$$

the neural network would essentially separate the input space, $\mathbf{R}^K$, by a hyperplane. Input vectors lying on one side of the hyperplane would be output as a 0, while input vectors on the other side of the hyperplane would be output as a 1. The hyperplane is determined by the weights and node offset value of the perceptron. If the hard limiter were replaced by the sigmoid, the decision region would become *soft.*

More complex decision regions can be formed by utilizing multiple hyperplanes. Decision regions can be formed by using a perceptron to form each hyperplane of a complex region. The output of each perceptron can then be fed into an AND gate — or, better yet, another perceptron with weights and an offset appropriately set to simulate an AND function. This leads to the concept of multi-layer perceptron neural networks.

Multiple-layer perceptron neural networks take the outputs of the perceptrons on a layer and use them as inputs to the next higher level of perceptrons (see Figure 1). Networks of this type are usually called feed-forward neural networks. As demonstrated in the above discussion, a single perceptron can only divide the decision space with a hyperplane. But it has been shown that a two-layer perceptron neural network can form any convex decision region [5]. A convex region is a region from which any two points can be connected by a line which lies entirely within the region. A third layer of nodes can allow the network to form any arbitrary decision region [5] (assuming enough nodes are allocated to the correct layers).

To form a desired decision region, the weights and node offset values for each node in each layer of a neural network must be specified. This would be a difficult task even if the decision region were known. But, for many problems, the decision region is not known because the statistical models of the data are not known. Training algorithms to form appropriate decision regions exist for perceptron neural networks. These algorithms typically present the training data to the network along with a desired response and the network weight values and node offset values are adjusted to force the actual network response towards the desired response. One such algorithm is the back-propagation algorithm. The back-propagation algorithm is a gradient search method (searching over $\mathbf{w}$ and θ), which minimizes the square error of the neural network outputs [6]. Note that the back-propagation algorithm requires the nodes to have sigmoidal nonlinearities.

III. The Discriminator

We start by defining the structure of our sequential discriminator. Our discriminator utilizes a test statistic of the form

$$T_n(\mathbf{Z}) = \sum_{j=1}^{n-K+1} \gamma(Z_j, Z_{j+1}, \ldots, Z_{j+K-1}). \tag{8}$$

A two threshold test is implemented, using the constants $\tilde{a}$ and $\tilde{b}$. So, upon obtaining a new data sample, Z_n, the discriminator computes the test statistic T_n. If T_n reaches $\tilde{b}$, then the discriminator chooses H_1 and terminates the test. If T_n drops to $\tilde{a}$, then the test terminates and the discriminator chooses H_0. If T_n lies between $\tilde{a}$ and $\tilde{b}$, then another sample Z_{n+1} is obtained, T_{n+1} is computed, and the entire test is repeated. This process continues until either a decision is made, or the N-th sample is reached. Upon obtaining the N-th sample, T_N is computed and a one-threshold test is performed. Obviously T_{K-1} is initialized to a value in the interval $(\tilde{a}, \tilde{b})$.

We now restrict the class of nonlinearities of the form
$\gamma(x_1, x_2, \ldots, x_K)$ to have a range with maximum absolute value of r. That is, we require

$$|\gamma(x_1, x_2, \ldots, x_K)| \leq r \text{ for all possible values of}$$

$$\text{the } K - \text{tuple } (x_1, x_2, \ldots, x_K). \tag{9}$$

This restriction leads to a suboptimal discriminator in general, but allows us to obtain a solution.

Now assuming that r, $\tilde{a}$, and $\tilde{b}$ are all specified constants, the structure of our test allows us to scale r, $\tilde{a}$, and $\tilde{b}$ to get a test with a nonlinearity with a maximum absolute value of 1. The newly scaled thresholds shall be denoted as a and b. This rescaling of r to 1 allows us to utilize a perceptron neural network with a sigmoid nonlinearity on its nodes in the following paragraphs.

To find the optimal nonlinearity within our class, we first consider the optimal paths that the test statistic T_n can take under each hypothesis. By optimal path we mean the path that T_n should take to minimize the number of samples needed to cross the

correct threshold under the appropriate hypothesis. Obviously the quickest path to reach a threshold is when the discriminator takes a step of magnitude 1 in the appropriate direction upon obtaining each new data sample. That is, for each new data sample, the test statistic under H_1 is incremented by $+1$, while the test statistic under H_0 is incremented by -1. If the data sequence $\{Z\}_{i=1}^{\infty}$ is obtained by sampling some continuous process with a uniform sampling period T, then the optimal path for T_n would lie on a straight line with slope $+\frac{1}{T}$ for H_1 and slope $-\frac{1}{T}$ for H_0. Figure 2 depicts these paths. Thus, for an ideal discriminator, (that is a discriminator which never makes mistakes and always uses the minimum number of samples possible), the statistics of the nonlinearity should be

$$E_1[\gamma(Z_1, Z_2, \ldots, Z_K)] = 1$$

$$E_0[\gamma(Z_1, Z_2, \ldots, Z_K)] = -1$$

$$Var_1[\gamma(Z_1, Z_2, \ldots, Z_K)] = 0$$

$$Var_0[\gamma(Z_1, Z_2, \ldots, Z_K)] = 0 \tag{10}$$

where E_i and Var_i denote expectation and variance respectively under hypothesis H_i.

We cannot expect a real discriminator to achieve the statistics of the above equations. However, we can choose the nonlinearity to minimize some performance measure, such as a mean squared error criterion of γ about its desired values. We show that the back-propagation algorithm can be used to minimize a related mean squared error criterion.

We form a nonlinearity by constructing a perceptron neural network with K inputs $x_1, x_2, \ldots, x_K$ and two outputs which are functions of the inputs (and the weights/offsets for each perceptron in the network), $o^1(x_1, x_2, \ldots, x_K)$ and $o^0(x_1, x_2, \ldots, x_K)$. To simplify the notation we denote the output nodes as o^1 and o^0. During training, the desired values of the output nodes are (1 0) for inputs from H_0 and (0 1) for inputs from H_1. Our notation $(x\ y)$ implies that $o^0 = x$ and $o^1 = y$. The nonlinearity, $\gamma(x_1, x_2, \ldots, x_K)$, is formed by

$$\gamma(x_1, x_2, \ldots, x_K) = o^1(x_1, x_2, \ldots, x_K) - o^0(x_1, x_2, \ldots, x_K),$$

or with simplified notation,

$$\gamma = o^1 - o^0. \tag{11}$$

We wish the nonlinearity to be such that γ is close to values of 1 for inputs from H_1 and -1 for inputs from H_0. We choose a performance measure which involves the mean squared error of o^1 and o^0 about their desired values for each hypothesis:

$$\tilde{S} = E_0 \left[(1-o^0)^2 + (0-o^1)^2\right] + E_1 \left[(0-o^0)^2 + (1-o^1)^2\right]. \tag{12}$$

We would like the weight and node offset values of each perceptron in our neural network to have values which minimize equation (12).

Recall that the back-propagation algorithm [6] is a gradient descent algorithm which minimizes the performance measure

$$\tilde{E} = \frac{1}{2} \sum_p \sum_j (t_p^j - o_p^j)^2 \tag{13}$$

where t_p^j is the desired output for node j associated with input pattern p, and o_p^j is the actual value of the output node j associated with input pattern p. Suppose we have P K-tuples from each hypothesis available for training the neural network. We also have $j = 0, 1$ for the two output nodes o^1 and o^0, respectively. We can rewrite (13) as

$$\tilde{E} = \frac{1}{2} \sum_{p=0}^{P-1} \{(1-o^0)^2 + (0-o^1)^2\} + \frac{1}{2} \sum_{p=P}^{2P-1} \{(0-o^0)^2 + (1-o^1)^2\} \tag{14}$$

where the first sum is over the H_0 training patterns and the second sum is over the H_1 training patterns. The problem of minimizing $\tilde{E}$ is equivalent to minimizing $\tilde{E}$ scaled by a constant. Thus minimizing (14) is equivalent to minimizing

$$\frac{2}{P}\tilde{E} = \frac{1}{P} \sum_{p=0}^{P-1} \{(1-o^0)^2 + (0-o^1)^2\} + \frac{1}{P} \sum_{p=P}^{2P-1} \{(0-o^0)^2 + (1-o^1)^2\}. \tag{15}$$

Now as $P \longrightarrow \infty$ we have

$$\frac{2}{P}\tilde{E} \longrightarrow E_0 \left[(1-o^0)^2 + (0-o^1)^2\right] + E_1 \left[(0-o^0)^2 + (1-o^1)^2\right] = \tilde{S}, \tag{16}$$

which is our desired performance measure. Consequently, the back-propagation algorithm is a reasonable algorithm to be utilized for our perceptron neural network nonlinearity. Thus, using this nonlinearity we can form the test statistic of equation (8).

Figure 3 shows the implementation of our test. The incoming data samples are passed through a tapped delay line. The K taps are the inputs to the perceptron neural network. The difference of the outputs of the neural network is formed and added to the test statistic T_j. The notation subscripts j correspond to the values associated with the jth data sample. The sample number j is compared to N. If j reaches N, then a one threshold test is performed (in this figure the threshold is 0.) If j is less than N, then a two threshold test is performed.

IV. Training

The neural network used in our sequential discrimination scheme operates on K-tuples $(Z_{K-1+j}, Z_{K-2+j}, \ldots, Z_j)$, which are formed
from the incoming data sequence $\{Z_j\}_{j=1}^{\infty}$ on which the discriminator must make a decision of H_1 or H_0. The neural network may have two or three layers of nodes, but it will always have two output nodes on the output layer.

The neural network is trained using the back-propagation algorithm and the training data set. The training data set consists of M sample paths (*i.e* sequences) of length N from each hypothesis. These training data are defined as $\zeta_{m,j}^{i}$, where $i = 0, 1$ denotes the hypothesis (H_1 or H_0), $m = 0, 1, \ldots, M-1$ denotes the sample path number, and $j = 0, 1, \ldots, N-1$ denotes the sample number. The desired responses for the neural network are (1 0) for H_0 and (0 1) for H_1. Our notation *(a b)* implies that the output node 0 outputs a and the output node 1 outputs b.

The training process proceeds as follows: The first K-tuple from the first sample path from H_0, $(\zeta_{0,0}^{0}, \zeta_{0,1}^{0}, \ldots, \zeta_{0,K-1}^{0})$, is presented to the neural network inputs. The back-propagation algorithm is performed using (1 0) as the desired output. Then the first K-tuple from the first sample path from H_1, $(\zeta_{0,0}^{1}, \zeta_{0,1}^{1}, \ldots, \zeta_{0,K-1}^{1})$, is presented to the neural network inputs. Back-propagation is performed with the desired response of (0 1). Then the second K-tuple from the first H_0 sample path, $(\zeta_{0,1}^{0}, \zeta_{0,2}^{0}, \ldots, \zeta_{0,K}^{0})$, is presented to the network for back-propagation. Then the second K-tuple from the first H_1 sample

path, $(\zeta^1_{0,1}, \zeta^1_{0,2}, \ldots, \zeta^1_{0,K})$, is presented to the network for back-propagation. When all the K-tuples (of ordered adjacent samples) from the first sample path for H_0 and H_1 have been exhausted, the process is repeated for the remaining until they have all been exhausted. Then the entire process is repeated until all sample paths have been presented to the network L times.

V. Results

The sequential neural network discriminators of the previous sections are evaluated. First, the training data is obtained by a computer simulation. After training, the discriminator is evaluated, again by simulated data. For simplicity, we take only one envelope sample per radar return pulse. Furthermore, we assume that the radar returns are correlated from pulse to pulse, and that the correlation structure modeled by a first order Markov process. The marginal pdfs of the simulated samples may be either lognormal or Rayleigh.

The Rayleigh processes are generated by underlying Gaussian processes (*i.e.* the in-phase and quadrature components.) We denote the envelope observations as $\{Z_i\}_{i=1}^{\infty}$. The Rayleigh envelope process is generated by

$$Z_i = \sqrt{X_i^2 + Y_i^2}, \ i = 1, 2, 3, \ldots \tag{17}$$

where $\{X_i\}_{i=1}^{\infty}$ and $\{Y_i\}_{i=1}^{\infty}$ are mutually independent Gaussian stationary first order Markov processes. This implies that $\{Z_i\}_{i=1}^{\infty}$ is also stationary and first order Markov. The underlying Gaussians are generated by

$$X_i = \rho X_{i-1} + \sqrt{1-\rho^2} V_i$$

$$Y_i = \rho Y_{i-1} + \sqrt{1-\rho^2} W_i, \ \text{for } i = 2, 3, \ldots. \tag{18}$$

with

$$X_1 = \sigma V_1$$

$$Y_1 = \sigma W_1 \tag{19}$$

where $\{V_i\}_{i=1}^{\infty}$ and $\{W_i\}_{i=1}^{\infty}$, are mutually independent sequences of i.i.d. (independent and identically distributed) zero mean/unit variance Gaussian random variables. σ is the

standard deviation of the underlying Gaussians, while ρ is the correlation coefficient for adjacent samples.

The correlation coefficient ρ is related to the decorrelation time τ in the following manner: τ is defined to be the time it takes for the correlation coefficient between the first sample and another sample to decrease by a factor of e^{-1}.

Our lognormal process is simulated by exponentiating an underlying Gaussian process:

$$Z_i = \exp(X_i + \mu), \;\; i = 1, 2, 3, \ldots \tag{20}$$

where X_i is generated in the same manner as equations (18) and (19). Unlike the Rayleigh processes which have underlying Gaussians with zero mean, the underlying Gaussians for the lognormal process may have a mean μ.

Table 1 lists the discrimination cases tested with the sequential neural network discriminator. For Case 1, the target's samples are from a lognormal marginal density, while the decoy's samples are Rayleigh. The means of both hypotheses are matched; the powers of both hypotheses are also matched. The decorrelation times are $\tau_1 = 0.130290$ and $\tau_0 = 0.013029$, indicating that the decoy's samples become uncorrelated ten times as fast as the target's samples do.

For Case 2, both hypotheses are Rayleigh. However, a 3dB power difference exists in favor of H_1. The decorrelation times are identical to Case 1. Both hypotheses are again Rayleigh for Case 3. However in Case 3, the marginal pdfs are identical for both pdfs (matched means and powers). The decorrelation times remain unchanged from Case 1 and Case 2.

Table 2 details the neural networks trained for each case. Net 1 was trained for Case 1. Net 2 was trained for Case 2, and Net 3 was trained for Case 3. All networks had two layers. Each network also had $K = 4$ inputs, and $N_0 = 16$ nodes on its first layer of nodes. The nets were trained with training data (generated by computer simulation) as described above. For each case, the training data consisted of $M = 50$ sample paths from each hypothesis. Each path consisted of $N = 1000$ samples. The number of times each sample path was presented to the network during training was $L = 100$. The gain constant in the back-propagation algorithm (see [6]) was set to 0.001. The column labeled $\tilde{S}$ in Table 2

represents the performance measure defined by equation (12). After the training process was completed, $\tilde{S}$ was estimated by computing the average squared difference between the desired net output and the actual output when the training data was again presented to the net. Recall, this measure is a mean squared error of the actual neural network output about the desired response.

After training the neural networks, they were inserted into the discriminator structure of Figure 3. The thresholds a and b were chosen by experimentation; a was set to -20 and b was set to 20. The initial value of the test statistic was set to 0. Computer simulated data were then fed to the discriminator simulations. $10,000$ simulated sample paths from each hypothesis for each case were presented to the discriminator for classification. Table 3 gives the results for each case. Table 3 also gives the results of [3] for Case 1 and 2. Optimal memoryless nonlinearities in [3] were derived from *known* probability density functions. One could expect optimal memoryless nonlinearities derived from estimated pdfs (*e.g.* estimated from the training data) to perform worse. [3] did not include results for Case 3; this was due to a limitation imposed by their performance measure, which became zero when the marginal pdfs under both hypotheses were identical. The columns P_1 and P_0 in Table 3 represent the measured probabilities of error under H_1 and H_0 respectively. The column labeled $E[n]$ contains the average number of samples required for the test to terminate.

VI. Conclusions

A scheme for utilizing perceptron neural networks in a discrimination scheme utilizing the test statistic of equation (3) was presented. This test statistic can be readily utilized in either a block or sequential tests. The neural network's training phase eliminates the impractical task of estimating high-order pdfs when designing a discriminator; consequently discriminators with memory (*i.e.* $K > 1$) are easily obtained.

Some results were presented for a sequential implementation. The discriminators using neural networks for their nonlinearities significantly out-performed the optimal memoryless discriminators of [3]. The discriminators constructed with neural networks made no classification errors in 10,000 trials from each hypothesis! These discriminators also used a significantly smaller expected number of samples to make their decisions than did the

discriminators of [3] (refer to Table 3)! Furthermore, the neural networks were able to converge to a nonlinearity for Case 3; [3] could not even consider Case 3 due to the fact that the marginal pdfs under both hypotheses were identical.

References

[1] H. Vantrees, *Detection, Estimation, and Modulation Theory, Part I*, John Wiley & Sons, New York, NY, 1968.

[2] D.W. Sauder and E. Geraniotis, "Optimal and Robust Memoryless Discrimination from Dependent Observations," *IEEE Trans. Information Theory,* Jan. 1991, to appear.

[3] E. Geraniotis, "Sequential Tests for Memoryless Discrimination from Dependent Observations," submitted to *IEEE Trans. Information Theory,*, 1989, under review.

[4] D.W. Sauder and E. Geraniotis, "Optimal One-Step Memory Nonlinearities for Signal Discrimination from Dependent Observations," appeared in *Proceedings of 1990 Conference on Information Sciences and Systems*, Princeton University, 1990.

[5] R. Lippmann, "An Introduction to Computing with Neural Nets," *IEEE ASSP Magazine,* vol. 4, pp 4-22, April 1987.

[6] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning Internal Representations by Error Propagation" in D.E. Rumelhart & J.L McClelland (Eds.), *Parallel Distributed Processing: Exploration in the Microstructure of Cognition, vol. 1: Foundations.,* MIT Press, 1986.

Case	pdfs H_1 vs H_0	Power Ratio H_1 vs H_0	Mean Ratio H_1 vs H_0	Decorrelation Times τ_1,τ_0
Case 1	Lognormal vs Rayleigh	0dB	0dB	0.130290, 0.013029
Case 2	Rayleigh vs Rayleigh	3dB	—	0.130290, 0.013029
Case 3	Rayleigh vs Rayleigh	0dB	—	0.130290, 0.013029

Table 1: Description of the Discrimination Test Cases

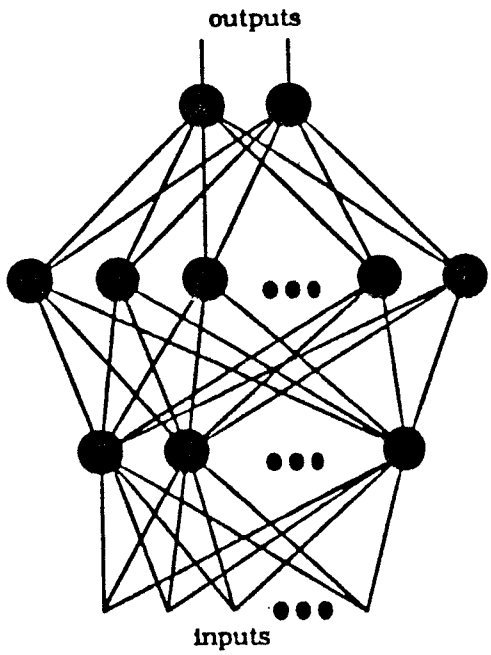


Figure 1. A Multiple Layer Perceptron Neural Network

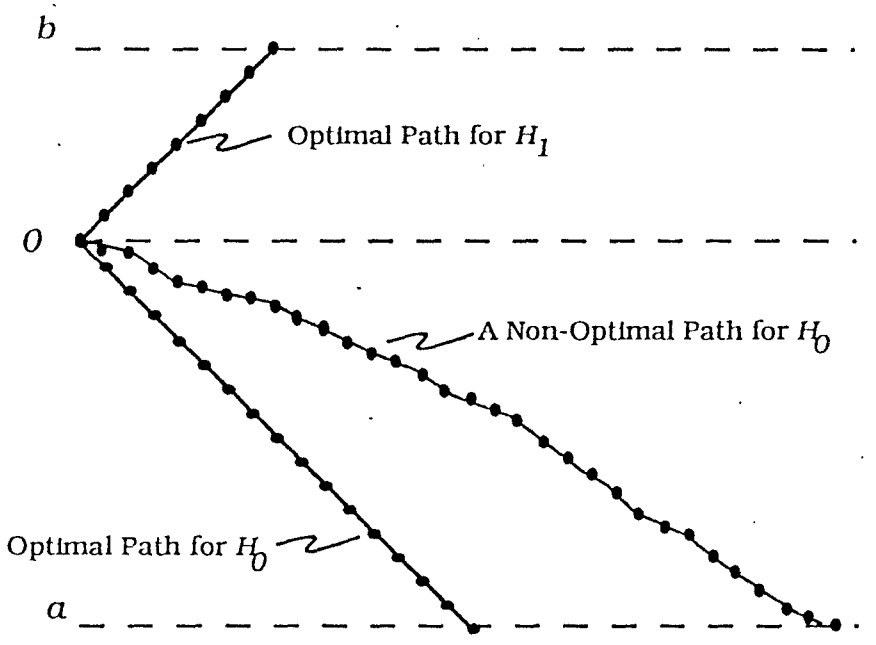


Figure 2. Optimal Paths for the Test Statistic

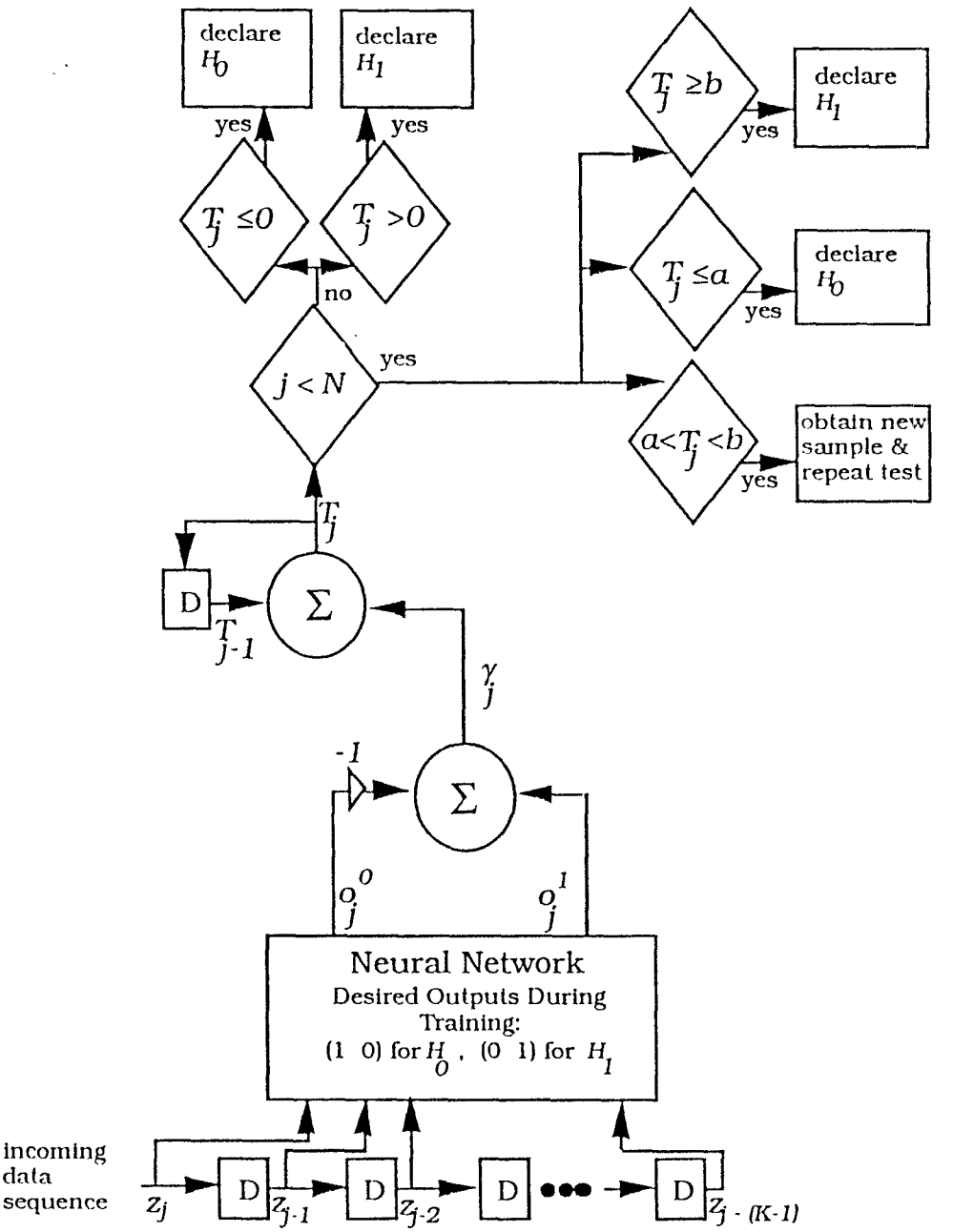


Figure 3. Discriminator Structure

Table 1: Description of the Discrimination Test Cases

Case	pdfs H_1 vs H_0	Power Ratio H_1 vs H_0	Mean Ratio H_1 vs H_0	Decorrelation Times τ_1,τ_0
Case 1	Lognormal vs Rayleigh	0dB	0dB	0.130290, 0.013029
Case 2	Rayleigh vs Rayleigh	3dB	—	0.130290, 0.013029
Case 3	Rayleigh vs Rayleigh	0dB	—	0.130290, 0.013029

Table 2: Neural Networks for Each Case

Case	Network	K	N_0	$\tilde{S}$
Case 1	Net 1	4	16	3.259×10^{-6}
Case 2	Net 2	4	16	4.075×10^{-6}
Case 3	Net 3	4	16	4.012×10^{-6}

Table 3: Results for 10,000 Sample Paths from Each Hypotheses

Case	Discriminator	P_1	P_0	$E[n]$
Case 1	Net 1	0	0	32
Case 1	Memoryless	.010	0	572
Case 2	Net 2	0	0	55
Case 2	Memoryless	.003	.002	2025
Case 3	Net 3	0	0	41