K. M. Passino, M. A. Sartori and P. J. Antsaklis, "Neural Computing for Information Extraction in Control Systems," Proc. of the 26th Allerton Conference on Communication, Control and Computing, pp. 1172-1180, Allerton, IL Sept. 28-30,1988.

Presented at the Twenty-Sixth Annual Allerton Conference on Communication, Control, and Computing
University of Illinois at Urbana-Champaign
28-30 September 1988

NEURAL COMPUTING FOR INFORMATION EXTRACTION IN CONTROL SYSTEMS

K.M. PASSINO, M.A. SARTORI, and P.J. ANTSAKLIS
Department of Electrical and Computer Engineering
University of Notre Dame
Notre Dame, IN 46556

Abstract

Neural computing offers massively parallel computational facilities for the classification of patterns. In this paper we use a certain type of neural network called the "multi-layer perceptron" to classify numeric data and appropriately assign symbols to the various classes. It is shown that there are several control applications where a numeric to symbolic conversion process can be performed by the perceptron. The conversion process results in a type of "information extraction" which is similar to what is called "data reduction" in pattern recognition. After introducing the idea of using the neural network as a numeric to symbolic converter, several applications are studied. First, the relevance of the results to autonomous control is discussed. The perceptron is used as a numeric to symbolic converter for a discrete event system controller supervising a continuous variable dynamic system. Also, it is shown how the perceptron can be used to implement a fault tree which provides useful information for failure diagnosis and control purposes.

1.0 Introduction

Neural computing involves the use of neural networks to perform computations in a massively parallel fashion. A neural network is a model of the interconnected network of neurons in, for instance, a human being. Researchers hope to emulate the highly efficient "computing" that humans perform with ease. Here, we use a certain neural network called the "multi-layer perceptron" to classify numeric data and appropriately assign symbols to the various classes. This classification is a form of information extraction from the given data. The classification is performed with massively parallel computing; hence, it is inherently faster than conventional sequential processing. Several applications to control systems are discussed including discrete event systems, autonomous control systems, and fault trees. Specific examples include the use of the perceptron to convert sensor data from a surge tank (the liquid level) to symbols that are used by a discrete event controller. Also, it is shown how the perceptron can be used to implement a fault tree for an aircraft.

It must be made clear that the goal of this paper is to introduce the idea of using the multi-layer perceptron as a numeric to symbolic converter to provide measurement information for specific control purposes. There is no contribution to the theory of neural networks or pattern recognition. The work involves the application of a particular neural network to problems often encountered in some of the more recent developments in control theory. In short, the results of this paper show one alternative for implementing the interface between numeric and symbolic information, a problem of significant importance.

Generally speaking, information extraction is the transformation of detailed information into abstract, generalized information; there is always a certain loss of information in the extraction process. Here, the detailed information is an n-dimensional vector of real numbers, the numeric data. The abstract information is a set of symbols representing regions in the n-dimensional space of real numbers. The information extraction process is performed here using the multi-layer perceptron. The type of information obtained from the extraction process studied here is of the form "signal 1 is greater than 0 and less than 5, and signal 2 is greater that -3 and less than 10". The information lost in our extraction process is the exact value of the real numbers. It is understood that there are other types of information extraction. For example, the type of information extraction studied here is limited in that it does not involve characterizing the *behavior* of the system generating the data. For instance, the results will not directly perform information extraction of the form "signal 1 is the derivative of signal 2". Consequently, the results obtained in this paper are not applicable to general information extraction problems. They only apply to a special class of information extraction problems that have been encountered in recent studies in control theory that are outlined below.

The process of information extraction via the neural network studied here is closely related to what has been called "pattern recognition". In fact, the multi-layer perceptron solves what are called "deterministic classification problems" [6]. The pattern recognition process is viewed as a three step "data reduction" process [6,5]:

(i) Sensors make *measurements* of certain variables in the environment,

(ii) *Feature extraction* is then used to obtain a more global, high level description of the measured data,

(iii) *Classification algorithms* are used to name the pattern.

The viewpoint can be taken that this complete process is performed by the multi-layer perceptron. The patterns sensed are points in the n-dimensional space of real numbers. Feature extraction entails

determining which region each element of the n-dimensional vector lies in. Classification involves taking logical combinations of these regions and attaching labels to them.

Rather than viewing the numeric to symbolic conversion process as a traditional pattern recognition problem, we prefer to view the multi-layer perceptron as the device which performs only step (i) (in the above process) of a more abstract recognition process in which the goal is to take some action based on the pattern detected. For step (i) of the more abstract process, the perceptron senses data in the environment and provides symbols as its measurements. These symbols are then used by a controller to make decisions on what to input into the plant. Clearly, the controller may need abstract, "symbolic" feature extraction and classification to complete the high level pattern recognition so it can make its decisions, but these problems are not studied here. A similar view of pattern recognition is taken in [23]. Note that the method used here does not employ the more complex pattern recognition techniques (e.g., statistical or syntactical) to recognize wider classes of patterns. For example, as indicated above it will not characterize the dynamical behavior of the environment over a period of time. For an overview of pattern recognition techniques see, for instance, [19].

Information extraction can be used in several different ways in control systems. The three applications studied here are described in more detail in Section 2. The first applications considered are certain discrete event systems. A sampling of references for this topic include [9,12,14,16,17,22,24]. Many others exist. Wonham and his student's work is particularly notable. In this paper we note that there are times when discrete event controllers can be used for the control of continuous variable systems. In these cases there is a need for a numeric to symbolic converter to provide the symbols to the controller. Next, for the case of autonomous control, there is a "symbolic/numeric interface" where high level, possibly intelligent controllers are used to control continuous systems [1]. The multi-layer perceptron can be used here to convert numeric information into more abstract symbolic information for the use in, for instance, an artificially intelligent planning system. Finally, the information can be extracted by the perceptron so that it is useful in higher level control decisions. For instance, the information provided by a fault tree on what the failure status of a plant is can be used to generate appropriate control actions so that the effects of the failures can be eliminated. It is shown here that the multi-layer perceptron can be used to implement a fault tree.

The theory of the multi-layer perceptron is outlined in Section 3. In Section 4, two examples are presented. First, an example of the use of the perceptron as a numeric to symbolic converter for a surge tank is given. Next, it is shown how the perceptron can be used for the implementation of a fault tree for an aircraft. Section 5 contains the references.

2.0 Information Extraction in Control Systems

In this section we discuss several ways in which information extraction is used in control theory. In practice, information is always extracted, in the modelling of processes for example. The extraction that we discuss here is the transformation of numeric data to symbolic information that is done dynamically in an on-line control system. There are three main applications to control theory which are discussed here.

The first is discrete event systems. Sometimes a plant is described by very complicated differential/difference equations. For some of these plants current control analysis and design techniques may be either very complicated, awkward, or simply inadequate. An alternative approach to analysis and design is to begin with a different model. In [14] the authors give an example of how to use a symbolic formalism, the nondeterministic finite state automaton, to describe a plant (a surge tank) that is normally described with a nonlinear differential equation. Using this description, a finite state controller was given and certain closed loop properties were verified. In these cases, for controller implementation, a numeric to symbolic converter is needed to convert the sensor data to symbols for use in the discrete event controller. The perceptron developed here is used to implement this converter.

The discrete event controllers considered in this paper will be referred to as "symbolic controllers" since they are developed using a symbolic formalism and have as inputs and outputs, symbols rather than numeric data. The situation where we have a symbolic controller controlling a continuous variable dynamic system is depicted in Figure 2.1. The symbolic controller uses Y_i (symbols) as inputs and a reference input, say R_i (symbols), and generates the control inputs to the plant U_i (symbols). The numeric/symbolic converter transforms the measurement data y(t) into symbols Y_i. The symbolic/numeric converter transforms the symbols U_i into numeric plant inputs u(t).

The numeric/symbolic converter can be thought of as analogous to analog to digital (A/D) converter in digital systems. In the one-dimensional case, the real number line (numeric data) is partitioned into regions associated with, say, binary numbers (symbols) and the values of the input are converted into the appropriate binary numbers. Indeed, there have been neural network implementations of a 4-bit A/D converter [21,7]. The work here generalizes that application. Besides performing a more general conversion

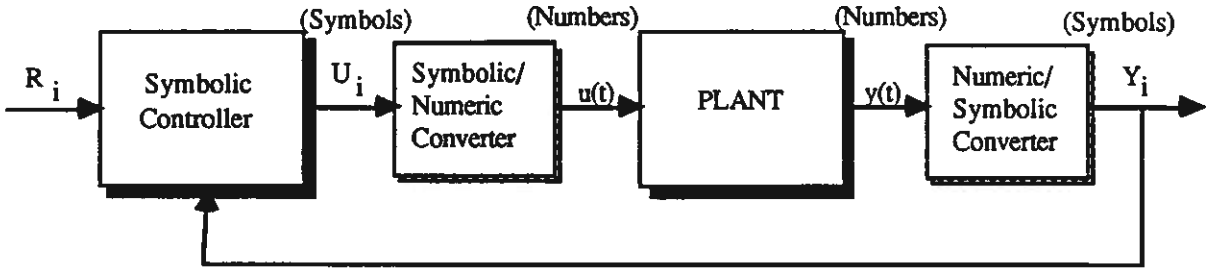


Figure 2.1 Symbolic/Numeric Conversion

process than A/D conversion, the perceptron can operate in an asynchronous mode in addition to a synchronous one. Note that the perceptron can be expected to perform numeric to symbolic conversion quickly. This is very important if it is to be placed in the control loop since delays tend to cause instability. The symbolic controller's control actions on the continuous plant can be translated to numeric inputs by a generalized method analogous to digital to analog (D/A) conversion. It may convert a symbol to a simple numeric value or to a sine wave, square wave, or some other more complex signal. This problem is not addressed here.

Autonomous controller architectures can be viewed as hierarchical with three main levels (see e.g., [1]). These are the management and organization level, the coordination level, and the execution level. The execution level contains the hardware and conventional control algorithms performing low level feedback control. The management and organization level has the interface to the operator and the intelligent learning and planning systems for supervising the actions taken at the lower levels. The coordination level provides a link between the management and organization level and the execution level. Autonomous controllers are used in the control of very complex systems that are normally hybrid. A "hybrid" system is one that contains dynamics that can be described by, for instance, differential or difference equations, and dynamics that are convenient to describe with some symbolic formalism. The controllers developed for such systems are also hybrid, with various numeric and symbolic components at each of the three levels. The numeric type algorithms are normally used at the lower levels of the architecture, the execution level, while the symbolic type algorithms are operating at the highest level, the management and organization level. There is then a symbolic/numeric interface in the autonomous controller much like the one in discrete event system theory discussed above. Consequently, the perceptron developed here will be useful in intelligent autonomous control applications.

The results here can also be used in the field of intelligent autonomous control called Expert Control. The "safety nets" in [2] use information extraction of ranges of variables for making high level control decisions in adaptive control. The results of this paper provide a method to produce the information for such control decisions. In [2] the authors propose to use an "expert controller", one that uses an expert system to emulate the actions that a human operator would perform to maintain an adaptive controller's operation. The human in the control loop is being emulated. It is then natural to use a perceptron to emulate the human's pattern classification abilities.

In general the multi-layer perceptron can be used to classify data so that control decisions can be made. The symbols attached to various regions of the n-dimensional space of real numbers may have a particular physical interpretation. The perceptron can provide failure information such as "signal 1 is in range 4 AND signal 3 is in range 5 OR signal 7 is in range 11". This information may be quite useful in making control decisions. For instance, the values of signals 1, 3, and 7 may indicate that a certain failure condition is occurring and that a specific control action ought to be taken so that the failure will not degrade system performance. Failure information of the type described above is typically generated by Fault Trees [4,3]. The *low level failures* in fault trees are characterized by regions of certain variables and fault tree *high level failures* by logical combinations of the low level failures. Here, the multi-layer perceptron determines which regions certain variables occur in and performs the appropriate ANDing and ORing to indicate what high level failure has occurred. In the following section, an outline of the theory of the multi-layer perceptron is given.

3.0 The Multi-Layer Perceptron for Information Extraction

The multi-layer perceptron is a feedforward neural network used for neural computing. The multi-layer perceptron considered here contains a *hidden layer* between the *input* and the *output layers* as illustrated in Figure 3.1. The input to the perceptron is the vector $u=[u_1\ u_2\ ...\ u_M]^t$ (where t denotes transpose), and it contains M continuous real valued elements. The vector $x'=[x'_1\ x'_2\ ...\ x'_{M1}]^t$, which contains M_1 binary elements, is the output of the input layer and the input to the hidden layer. The vector $x''=[x''_1\ x''_2\ ...\ x''_{M2}]^t$, which contains M_2 binary elements, is the output of the hidden layer and the input to the output

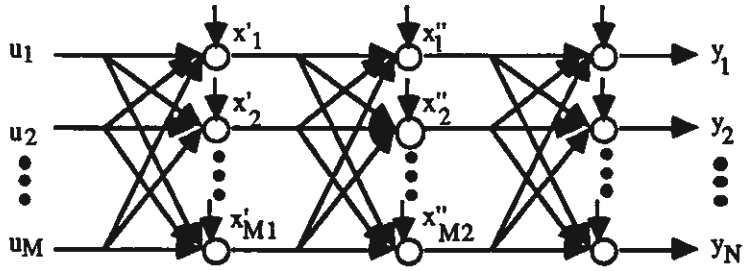


Figure 3.1 The Multi-Layer Perceptron

layer. The vector $y=[y_1\ y_2\ ...\ y_N]^t$, which contains N binary elements, is the output of the perceptron. In Figure 3.1, the *nodes* are denoted with circles and the *biases* with arrows that point downward. The *weights* are denoted by all of the other arrows that are between u and the input nodal layer, x' and the hidden nodal layer, and x" and the output layer. Each node produces at its output a summation of its weighted inputs and its bias which is passed through a threshold nonlinearity. The result is a binary output for each layer. Two typical threshold nonlinearities are illustrated in Figure 3.2.

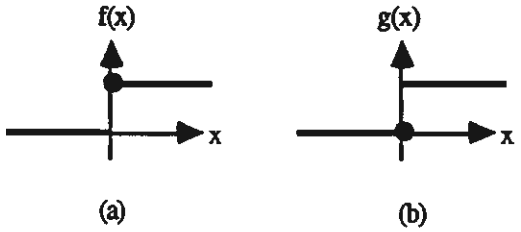


Figure 3.2 Threshold Nonlinearities for the Multi-Layer Perceptron

The multi-layer perceptron is commonly used for "pattern classification". In the classification problem one desires to identify certain "decision" regions that the input vector u lies in. Examples of dividing a two dimensional input space into decision regions are illustrated in Figure 3.3.

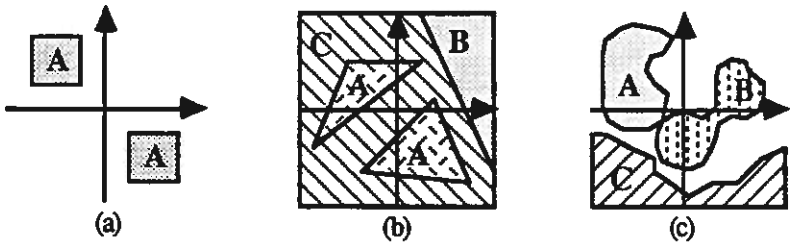


Figure 3.3 Examples of Two Dimensional Decision Regions

An input pattern u is presented to the perceptron. After processing, the output vector y represents certain decision regions. In particular, if u is a member of one of the decision regions, an appropriate element of the output vector y becomes a one (HI). If the input pattern u is not a member of a particular decision region, an appropriate element of the output vector y becomes a zero (LO).

The weights and biases of the multi-layer perceptron depend on the decision regions to be identified. The technique used to determine the weights and biases for the multi-layer perceptron here will be referred to as the "Harvey Method". It is based on the results presented in [8] and is outlined here. The Harvey Method can be used most effectively if the input space can be divided into decision regions composed of hypercubes, as in Figure 3.3(a), or hyperplanes, as in Figure 3.3(b). Using the Harvey Method, not only are the weights and biases determined but also the actual interconnections between the nodal layers. As with the weights and biases, the interconnections also depend on the decision regions to be identified. The three layers of the multi-perceptron, as shown in Figure 3.1, are named (from left to right) the input/plane layer, the AND layer, and the OR/output layer. The input/plane layer consists of nodes representing the hyperplanes dividing the input space. For example, in Figure 3.3(a) region A needs eight one-input input/plane nodes since the region is constrained by eight different lines. The AND layer consists of one AND node for each decision region defined by more than one hyperplane. For example, in Figure 3.3(a) region A needs two four-input AND nodes, each to denote that u is below one line, to the left of another line, above one line, and to the right of still another line. In other words, each four input AND node will

indicate that u lies in one of the two boxes that define region A. The OR/output layer consists of nodes used to associate disjoint decision regions. For example, in Figure 3.3(a) region A needs one two-input OR/output node to indicate that u lies in one of the two boxes. Using the Harvey Method, the weights depend on which side of the plane the decision region lies, and the biases depend on the locations of the decision regions. As an alternative, the weights and biases could be determined by passing sample decision region data through the multi-layer perceptron and updating the weights and biases based on a training algorithm explained in [8].

There are numerous other methods used to determine the weights and biases. For instance, the Back-Propagation Training Algorithm which is presented in [18,15] can be used for perceptrons similar to the one shown in Figure 3.1, but with fully interconnected nodal layers. This algorithm may be particularly useful if the input space must to be divided into decision regions which are separated by curved borders, as in Figure 3.3(c). In [10] three other training procedures are introduced and compared to the Back-Propagation Training Algorithm. In [11], an alternative to the Back-Propagation Training Algorithm, called the "Selective Update Back-Propagation Algorithm", is introduced and shown to work in cases where the Back-Propagation Training Algorithm will not. Through the presentation of input data, the multi-layer perceptron's output, and the desired output, these algorithms train the weights and the biases until they stabilize. Although the Back-Propagation Training Algorithm seems to work for most cases, convergence of the algorithm has not yet been proven for the fully interconnected multi-layer perceptron [15,18]. Convergence for the lightly interconnected multi-layer perceptron is considered in [20].

4.0 Examples of Information Extraction In Control Systems

In this section two examples of how information extraction can be performed via the multi-layer perceptron are given. The first example, from chemical process control, involves a liquid holding "surge tank" similar to the one studied in [14]. The tank is depicted in Figure 4.1 below.

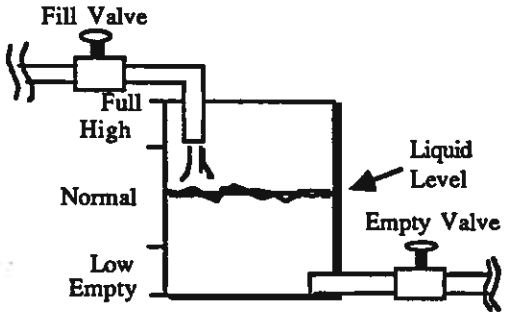


Figure 4.1 Surge Tank

The empty valve is unpredictably opened by some user and the controller turns the fill valve on and off so as to keep the tank from becoming empty or full. The surge tank can be described with the first order nonlinear differential equation [13]:

$$\frac{dh(t)}{dt} = \frac{F_i(t)}{\rho A} - \frac{kc(t)\sqrt{h(t)}}{\rho A}$$

where ρ=99.8, the fluid density of the liquid, A=1, the cross sectional area of the tank, h(t) is the height of the liquid in the tank, and k=1000, a physical constant. The value of c(t) represents the variable outflow pipe cross sectional area. The unpredictable user is modelled as $c(t)=.1c_o(t)$, where $c_o(t)$ is a random variable uniformly distributed on 0 to 1. The value of $F_i(t)= 100\sqrt{3}F(t)$, where F(t) is either 1 (indicating that the fill valve is on), or 0 (indicating that the fill valve is off). Notice that with this choice if the input valve is open the liquid level will stay the same or rise.

The height of the liquid in the tank is discretized into 5 regions which are represented with the symbols y_i, i=1,2,3,4,5. The case when F(t)=1 will be represented with the symbol u_1 and F(t)=0 with the symbol u_0. The symbolic/numeric converter is simply a device that outputs 1 when it has as an input u_1 and 0 when its input is u_0. The controller which is the same as in [14] is given by:

Controller					
Controller Input	y_1	y_2	y_3	y_4	y_5
Controller Output	u_1	u_1	u_1	u_0	u_0

Next, the numeric/symbolic converter must be specified. Let h_1=0 (empty) and h_4 represent the case when the tank is completely full. Also let h_2 and h_3 represent two particular levels in the tank which distinguish, respectively, a low liquid level from a normal one and a normal level from a high one. The table below gives the necessary information to specify the classification problem.

Numeric/Symbolic Converter					
Numeric/Symbolic Converter Output	y_1	y_2	y_3	y_4	y_5
Plant Output Regions	$h(t)=h_1$	$h_1<h(t)<h_2$	$h_2\leq h(t)\leq h_3$	$h_3<h(t)<h_4$	$h(t)=h_4$

The multi-layer perceptron designed using Harvey's Method is illustrated in Figure 4.2.

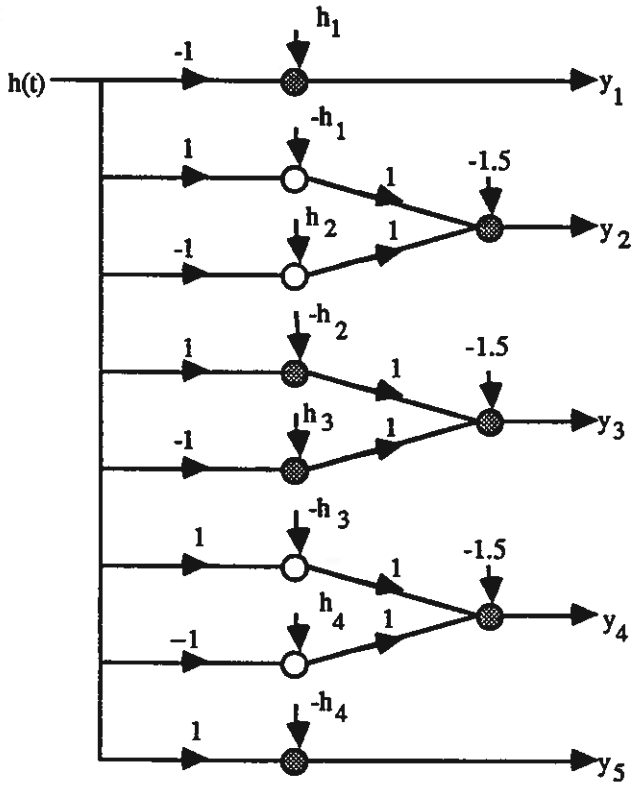


Figure 4.2 The Multi-Layer Perceptron for the Surge Tank

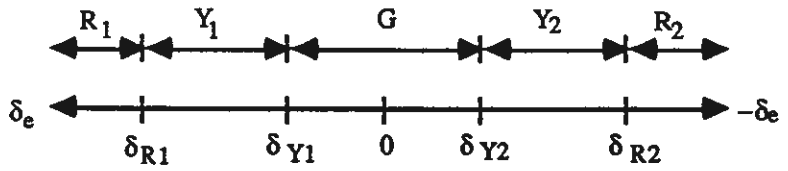
The arrows indicate amplifiers. The shaded and unshaded circles indicate a summing of the inputs to the circle (arrows pointing towards the circle) and that the sum is passed through one of the two nonlinearities shown in Figure 3.2. The shaded circles refer to Figure 3.2(a) and the unshaded circles refer to Figure 3.2(b).

The object of the controller studied in [14] was to stabilize the height of the liquid in the surge tank around a particular level. In particular, it was desired that the steady state behavior of the plant entail only a sequence of y_3 and y_4 alternating. When the plant, the numeric/symbolic converter, the controller, and the symbolic/numeric converter are connected together as in Figure 2.1 and simulated, the output of the plant behaved as desired. The plant was given two initial conditions, completely empty, and at level y_3. Each time, the controller opened the fill valve and then closed the fill valve when the level rose to y_4. The input valve was opened again when the lower level fell to y_3 and closed again when the level rose to y_4. The system continued in this fashion with the controller attempting to keep the level just above y_3. Consequently, the multi-layer perceptron successfully implemented the numeric/symbolic converter for the surge tank and its symbolic controller.

The second example involves the implementation of a fault tree for an aircraft. The multi-layer perceptron is used to indicate the failure modes which depend on the aircraft's inputs and outputs. The aircraft's input vector u and the output vector y are:

$$u = \begin{bmatrix} \delta_e \\ \delta_t \end{bmatrix} \begin{matrix} \text{elevator (deg)} \\ \text{thrust (deg)} \end{matrix} \qquad y = \begin{bmatrix} q \\ \theta \\ \eta_z \end{bmatrix} \begin{matrix} \text{pitch rate (deg/sec)} \\ \text{pitch angle (deg)} \\ \text{load factor (g)} \end{matrix}$$

Four failure modes are considered here. To do so, each input and output is discretized into five regions with four boundaries associated with the real number line. For example, the elevator δ_e is discretized as follows:



where the G (for Green) region denotes an area of safe operation, the Y_1 and Y_2 (for Yellow) regions denote areas of warning, and the R_1 and R_2 (for Red) regions denote areas of unsafe operation. The table below defines the five regions for the elevator δ_e.

Elevator δ_e Regions					
Elevator Region	R_1	Y_1	G	Y_2	R_2
Elevator Range	$\delta_e \le \delta_{R1}$	$\delta_{R1} < \delta_e \le \delta_{Y1}$	$\delta_{Y1} < \delta_e < \delta_{Y2}$	$\delta_{Y2} \le \delta_e < \delta_{R2}$	$\delta_{R2} \le \delta_e$

The four other aircraft parameters are discretized in a similar manner.

Using the defined regions for the parameters, four failure modes for the aircraft are identified as follows:

1) load factor is in region R_2 ($\eta_z \in R_2$)
2) load factor is in region Y_2 ($\eta_z \in Y_2$)
3) load factor is in region Y_2 and elevator is in region Y_1 ($\eta_z \in Y_2$ and $\delta_e \in Y_1$)
4) (pitch rate is in Y_1 and pitch angle is in Y_1) or (pitch rate is in Y_2 and pitch angle is in Y_2) [($q \in Y_1$ and $\theta \in Y_1$) or ($q \in Y_2$ and $\theta \in Y_2$)]

For the four failure modes decision regions can be defined and a multi-layer perceptron can be designed using the Harvey Method. The perceptron's inputs are the inputs and outputs of the aircraft and the perceptron's outputs are the four fault modes. For the first mode, the output is HI if the load factor η_z is in region R_2. Let $\Leftrightarrow$ denote equivalence. Notice that

$$\eta_z \in R_2 \Leftrightarrow \eta_z \ge \eta_{R2} \Leftrightarrow \eta_z - \eta_{R2} \ge 0$$

Using the last equation, the multi-layer perceptron for the first fault mode is shown at the top of Figure 4.3. For the second fault mode, the output should be HI if the load factor η_z is in region Y_2. Notice that

$$\eta_z \in Y_2 \Leftrightarrow \eta_{Y2} \le \eta_z < \eta_{R2} \Leftrightarrow \begin{cases} \eta_z \ge \eta_{Y2} \Leftrightarrow \eta_z - \eta_{Y2} \ge 0 \\ \text{and} \\ \eta_z < \eta_{R2} \Leftrightarrow -\eta_z + \eta_{R2} > 0 \end{cases}$$

Using the last equations, the multi-layer perceptron for the second fault mode is shown in the middle of Figure 4.3. For the third fault mode, the output should be HI if the load factor η_z is in region Y_2 and the elevator δ_e is in region Y_1. Notice that

$$\eta_z \in Y_2 \Leftrightarrow \eta_{Y2} \le \eta_z < \eta_{R2} \Leftrightarrow \begin{cases} \eta_z \ge \eta_{Y2} \Leftrightarrow \eta_z - \eta_{Y2} \ge 0 \\ \text{and} \\ \eta_z < \eta_{R2} \Leftrightarrow -\eta_z + \eta_{R2} > 0 \end{cases}$$

and

$$\delta_e \in Y_1 \Leftrightarrow \delta_{R1} < \delta_e \le \delta_{Y1} \Leftrightarrow \begin{cases} \delta_e > \delta_{R1} \Leftrightarrow \delta_e - \delta_{R1} > 0 \\ \text{and} \\ \delta_e \le \delta_{Y1} \Leftrightarrow -\delta_e + \delta_{Y1} \ge 0 \end{cases}$$

Using the last equations, the multi-layer perceptron for the third fault mode is shown in Figure 4.3.

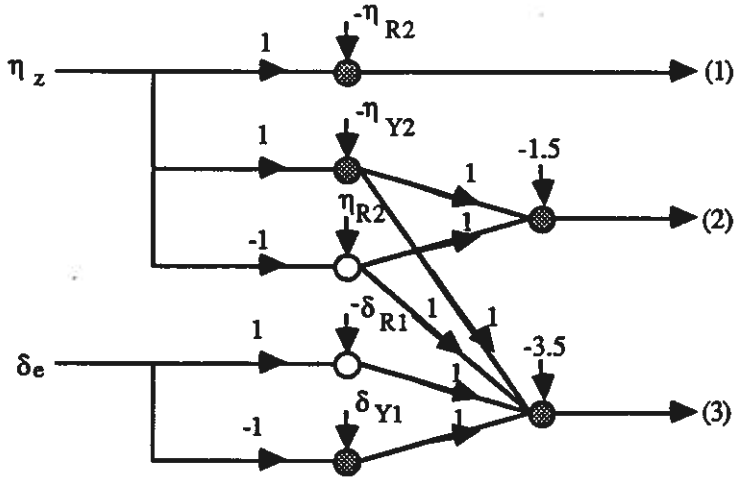


Figure 4.3 The Multi-Layer Perceptron for the First (1), Second (2), and Third (3) Fault Modes

For the fourth fault mode, either the pitch rate is in region Y_1 and the pitch angle is in region Y_1 or the pitch rate is in region Y_2 and the pitch angle is in region Y_2. Notice that

$$\begin{cases} q \in Y_1 \Leftrightarrow q - q_{R1} > 0 \text{ and } -q + q_{Y1} \geq 0 \\ \text{and} \\ \Theta \in Y_1 \Leftrightarrow \Theta - \Theta_{R1} > 0 \text{ and } -\Theta + \Theta_{Y1} \geq 0 \end{cases}$$

or

$$\begin{cases} q \in Y_2 \Leftrightarrow q - q_{R2} > 0 \text{ and } -q + q_{Y2} \geq 0 \\ \text{and} \\ \Theta \in Y_2 \Leftrightarrow \Theta - \Theta_{R2} > 0 \text{ and } -\Theta + \Theta_{Y2} \geq 0 \end{cases}$$

Using the last equations, the multi-layer perceptron for the fourth fault mode is shown in Figure 4.4.

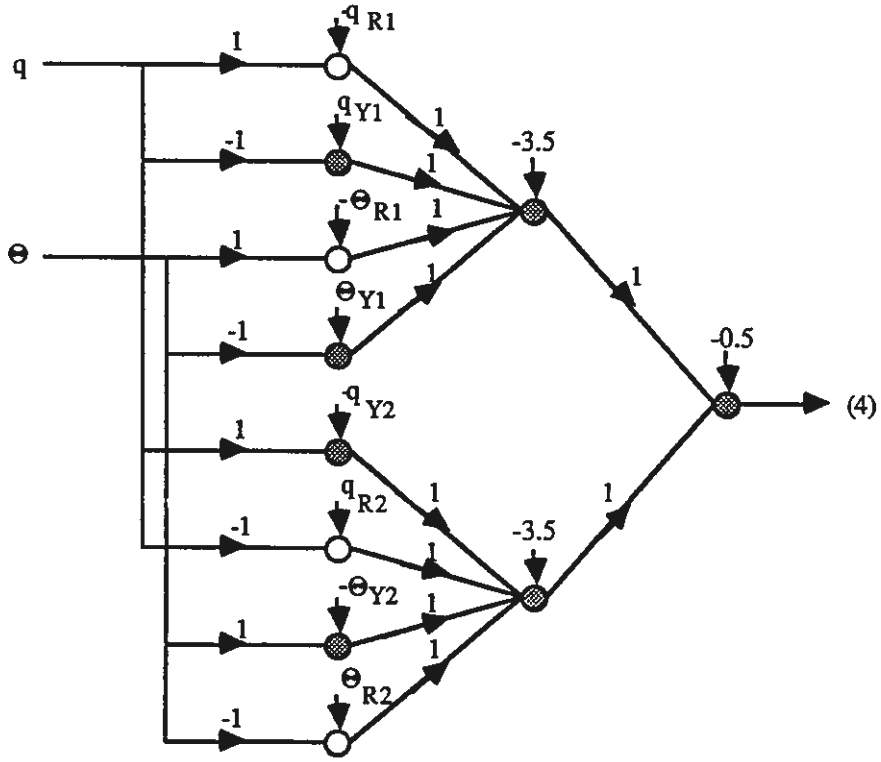


Figure 4.4 The Multi-Layer Perceptron for the Fourth (4) Fault Mode

The aircraft and the multi-layer perceptron which implemented the fault tree were simulated. The multi-layer perceptron appropriately identified all of the fault modes.

Acknowledgements:

The authors gratefully acknowledge the partial support of the Jet Propulsion Laboratory under Contract No. 957856 and the Army Research Office under Contract No. DAAL 03-88-K-0144.

5.0 References

[1] Antsaklis P.J., Passino K.M., Wang S.J., "Autonomous Control Systems: Architecture and Fundamental Issues", Proc. of the American Control Conf. pp. 602-607, Atlanta, June 1988.

[2] Astrom K.J., et al, "Expert Control", Automatica, Vol. 22, No. 3, pp. 277-286, 1986.

[3] Barlow R.E., et al, Reliability and Fault Tree Analysis, SIAM, Philadelphia, 1975.

[4] Battelle Columbus Division, Guidelines for Hazard Evaluation Procedures, American Institute of Chemical Engineers, NY, 1985.

[5] Forsyth R., Rada R., Machine Learning, Wiley, NY, 1986.

[6] Fu K.S., Sequential Methods in Pattern Recognition and Machine Learning, Academic Press, NY, 1968.

[7] Gray D.L., Synthesis Procedures for Neural Networks, Master's Thesis, Dept. of Electrical and Computer Eng., University of Notre Dame, Notre Dame, IN, July 1988.

[8] Harvey R.L., "Multi-layer Perceptron Theory", Dept. of Electrical and Computer Engineering Seminar, University of Notre Dame, March 21, 1988

[9] Ho Y.C., Li S., "Extensions of Infinitesimal Perturbation Analysis", Trans. on Automatic Control, Vol. 33, No. 5, pp. 427-438, May 1988.

[10] Huang W.Y., Lippmann R.P., "Neural Net and Traditional Classifiers", Presented at the IEEE Conference on Neural Information Processing Systems - Natural and Synthetic, Denver, CO, November 1987.

[11] Huang S.C., Supervised Learning with a Selective Update Strategy for Artificial Neural Networks, Master's Thesis, Dept. of Electrical and Computer Engineering, University of Notre Dame, Notre Dame, IN, Aug. 1988.

[12] Inan K., Varaiya P., "Finitely Recursive Process Models for Discrete Event Systems", IEEE Transactions on Automatic Control, Vol. 33, No. 7, pp. 626-639, July 1988.

[13] Johnson A., Process Dynamics Estimation and Control, Peregrinus Pub., London, 1985.

[14] Knight J.F., Passino K.M., "Decidability for Temporal Logic in Control Theory", Proceedings of the Twenty-Fifth Annual Allerton Conference on Communication, Control, and Computing, Vol. I, pp. 335-344, IL, 1987.

[15] Lippmann R.P., "An Introduction to Computing with Neural Networks", IEEE ASSP Magazine, pp. 4-22, April 1987.

[16] Ostroff J. S., Real-Time Computer Control of Discrete Systems Modelled by Extended State Machines: A Temporal Logic Approach, PhD Dissertation, Report No. 8618, Dept. of Elect. Eng., Univ. of Toronto, Jan. 1987.

[17] Passino K.M., Antsaklis P.J., "Branching Time Temporal Logic for Discrete Event System Analysis", Proceedings of the Twenty-Sixth Annual Conference on Communication, Control, and Computing, Sept. 1988.

[18] Rumelhart D.E., Hinton G.E., Williams R.J., "Learning Internal Representations by Error Propagation", in Rumelhart D.E., McClelland J.L. (Eds.), Parallel Distributed Processing: Explorations in the MicroStructrure of Cognition, Vol. 1: Foundations, MIT Press (1986).

[19] Shapiro S.C., ed., Encyclopedia of Artificial Intelligence, Volume 2, Wiley, NY, 1987.

[20] Shrivatava Y., Dasgupta S., "Convergence Issues in Perceptron Based Adaptive Neural Network Models", Proceedings of the Twenty-Fifth Annual Allerton Conference on Communication, Control, and Computing, Vol. II, pp. 1133-1141, IL, 1987.

[21] Tank D.T., Hopfield J.J., "Simple "Neural" Optimization Networks: An A/D Converter, Signal Decision Circuit, and a Linear Programming Circuit", IEEE Transactions on Circuits and Systems, Vol. CAS-33, No. 5, pp. 533-541, May 1986.

[22] Thistle J. G., Wonham W. M., "Control Problems in a Temporal Logic Framework", International Journal Control, Vol. 44, No. 4, pp. 943-976, 1986.

[23] Wilson R., "Is Vision a Pattern Recognition Problem?", in Kittler J. (Ed.), Pattern Recognition, Lecture Notes in Computer Science 301, Springer-Verlag, 1988.

[24] Wonham W.M., Ramadge P.J., "Modular Supervisory Control of Discrete-Event Systems", Mathematics of Control, Signals, and Systems, Vol. 1, pp. 13-30, 1988.