AN INVESTIGATION OF NEURAL NETWORKS FOR F-16 FAULT DIAGNOSIS: I. System Description

Richard J. McDuff[†]
Patrick K. Simpson[†]
David Gunning[‡]

We will report results of our ongoing research exploring the use of artificial neural networks (ANNs) for F16 flight line diagnostics. ANN's are based upon the methods of information processing that might be used in neural circuitry rather than those of a conventional computer. A significant aspect of neural networks is their ability to develop associations simply from examples. ANN's hold the promise of solving difficult logistics problems such as: multiple fault diagnosis, prognostication, changing configurations and environments, and inaccurate diagnosis attributable to incomplete and/or flawed rules. We tested three representative ANN's to see which type worked best for our problem. We chose back propagation (BPN) and counterpropagation (CPN) because they are considered to be two of the more promising pattern matching paradigms. The binary adaptive resonance theory I (ART1) was picked because it learns faster than CPN or BPN and has on-line adaption (i.e. does not have to be totally retrained every time a new pattern is discovered). On-line adaption is a powerful attribute, allowing new associations to be immediately incorporated into the knowledge base on the flight line or wherever needed. This paper explains the advantages and drawbacks to each network tested and describes how we trained them using historical flight line data. Of the three ANN's examined ART1 proved to be the most appropriate and was able to produce multiple symptom to multiple fault diagnoses.

1. INTRODUCTION

There are several problems that plague existing military diagnostic systems. Multiple fault diagnostics — one symptom leads to several faults, many symptoms lead to one fault, or many symptoms lead to many faults — is currently being performed with systems that are brittle and slow. Because of the need for time-critical flight-line response, available symptom data is either misinterpreted or unused — often leading to the incorrect removal of a system's component. Every new weapon system must develop its own diagnostic system, resulting in slowly developed and costly systems. Lastly, when a weapon system goes through a configuration change, a costly process of diagnostic system development for the new configuration becomes necessary to keep the diagnostic system up to date.

We are examining the use of neural networks [1] as a potential method of solving these maintenance and diagnostics problems. We have approached diagnostics from a pattern matching perspective in that we have constructed an input pattern from symptoms and matched that symptom pattern to an appropriate output pattern that corresponds to the fault that occurred. An operational sketch of this approach is shown in **Figure 1**.

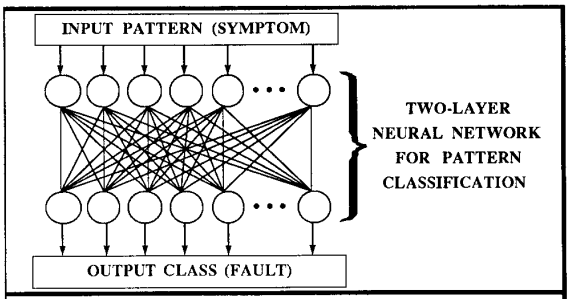


FIGURE 1: An illustration of the neural network approach to diagnostics. Input patterns (symptoms) are passed to a two-layer feedback neural network that determines which class the input belongs within. Once the input class is determined the outputs (faults) associated with that class are displayed. The most sophisticated neural network pattern classifiers (Adaptive Resonance Theory networks) will automatically add new classes when they are needed.

After an extensive comparative analysis of several key neural network paradigms, we have developed a neural network-based diagnostics system that addresses each of the previously stated problems. This system is targeted for eventual integration with the Air Force Human Resources Lab's Model-based Diagnostic Aiding System (MDAS). These preliminary results hold the potential promise for a generic weapon system diagnostic tool for any arbitrary weapon platform produced by any branch of the service (i.e. M1A1, F-16, ATF, ATA, SSN-21, etc.).

2563-1/ /0000-0351 01.

Our efforts sought to answer eight key questions:

(1) How should the training data for the networks be prepared?

(2) What is the most accurate way of representing the data?

(3) Which neural networks are most appropriate for these problems?

(4) What is the proper hardware to use?

(5) How well did the chosen network perform?

(6) How can we maximize the speed of this network?

(7) What is the optimal user interface design?

and

(8) How can we provide multiple fault output probabilities that emulate an MDAS probability table output?

2. DATA ACQUISITION, PREPARATION, AND REPRESENTATION

2.1 DATA ACQUISITION

The database used for training the neural networks is contained in the F-16 *Tactical Interim CAMS And REMIS Reporting System* (TICARRS) [2]. This database was acquired from the initial flight-line debriefing and the corrective actions taken. Each database entry contains information for each discrepancy (symptom-fault pair). The TICARRS database was transformed into a neural network training set that consisted of binary-valued symptom-diagnosis pairs.

The TICARRS database we received was not complete and contained many incorrectly entered discrepancies. In addition, many database entries did not contain a sufficient symptom description, often omitting important information such as the state of the plane at fault time and any intermediate tests that were conducted that led to a successful diagnosis. Because of this poor set of data, it became imperative that the neural network was able to learn and classify symptoms with inherently noisy and missing information.

2.2 DATA PREPARATION

Although the TICARRS database has problems, it is the only source of actual F-16 symptom-diagnosis data (and a good representation of what can be expected from any modern weapon system). A large portion of the time spent on this project was dedicated to data preparation. We created a neural network training set by filtering through the TICARRS database and correcting or discarding unsuitable entries.

From the TICARRS database we chose three key fields to represent the symptoms:

(1) MFL — Maintenance Fault List codes;

(2) FRC — Fault Reporting Codes; and

(3) WD — When Discovered codes.

A combination of these three fields is used as the input (symptoms) for the neural network. The MFL code is the only symptom acquired directly from the plane. The FRC code, when recorded out to the sixth digit, can represent a specific pilot complaint and is, therefore, useful symptom information. Sometimes, however, the FRC code is derived from the MFL code and is therefore redundant. The WD code is not a "formal" symptom, but is associated with the problem being diagnosed, therefore it is used. The remaining fault information, such as the air force base, the aircraft serial number, etc. was not used because of the data analysis time constraints, although, in a fielded system it might be beneficial.

Also, from the TICARRS database we chose two key fields to represent the fault that occurred:

(1) WUC — Work Unit Code; and

(2) AT — Action Taken (AT) code.

A four digit WUC specifies which Line Replaceable Unit (LRU) was involved, and a five digit WUC indicates which Shop Replaceable Unit (SRU) was involved. The AT code signifies what type of action was taken on the unit represented by the WUC. The narratives in both the discrepancy (symptom) and action taken (fault) portions of each report sometimes contain useful information but, since these are not formally or consistently encoded, they are not used. A complete description of the symptom and fault codes found in the TICARRS database is found in **Table 1**.

We acquired six months of fire control system data and prepared it for neural network training. From an initial set of 5,182 TICARRS entries, only 1,662 usable entries were found to be usable. We divided the usable discrepancies (i.e. the historical data) into two groups: the training set — a group of all but the last three weeks of database entries (1464 entries), and a test set — the last three weeks of database entries (198 entries). Although each of the database entries is usable, each entry is not necessarily accurate. In many instances the repair listed in the database was not what actually fixed the problem. Fortu-

MFL - consists of three letters followed by three numbers. There are 24 possible three letter combinations used in our system and so the vector has 24 on/off bits for them. There are a maximum of 341 three digit numbers and so the vector has 341 on/off bits for each. There are easily a thousand MFL codes. *Example MFL:* FCR 003
FRC - consists of 8 digits of which we use only the middle 4. The middle four digits consist of a number pair and then two separate digits. There are 100 possible two digit number combinations for the pair and so the vector has 100 possible on/off bits. There are 27 on/off bits each for the two separate letter or zero digits. There are over a thousand FRC codes possible. *Example FRC:* 9463AE00
WD - consists of a single digit which can either be a letter or a number. There are 36 possible letters or digits and so there are 36 on/off bits for the WD. *Example WD:* D
WUC - consists of a 5 or 6 digit number separated into a 2 digit number followed by 3 digits which can either be a letter or a number. The 2 digit number and the 3 single digits were each stored as is in the output vector. *Example WUC:* 74AKB
AT - consists of a single digit which can either be a letter or a number and was stored as is in the output vector. *Example AT:* R

TABLE 1: DESCRIPTION OF TICARRS DATABASE ENTRIES

nately this was not a problem, our neural network approach is able to handle such noise inherently.

2.3 DATA REPRESENTATION

2.3.1. Choosing a Representation Scheme

One of the primary difficulties with choosing a neural network pattern recognition approach to diagnostics is data representation. Because neural networks expect fixed length patterns (i.e. vectors) to be presented during training and use, a considerable amount of careful thought and planning is required to develop input (symptom) and output (fault) representations.

Three representation schemes were examined for our data: (1) variable-unit representations, (2) intermediate-unit representations, and (3) value-unit representations [3]. The first scheme (variable-unit representation) represents each symptom, or action taken, by a single neural network processing element (or vector component), whose value is the symptom code. Using this scheme a processing element (PE) in a neural network's input layer represents *all* MFL numbers (1000+) and its value would represent the MFL's ordinality. As an example, the MFL number 312 would be represented by a PE value of 312. An advantage to this representation is that it eliminates the need for an extensive analysis of the data to find all possible inputs and outputs. This representation also requires a smaller input and output vector dimensionality which will speed up the compute time for each pattern presentation. Conversely, such a representation creates a more difficult mapping because the low dimensional pattern space is often linearly inseparable. The more nonlinear the mapping, the more difficult it is to acquire an accurate mapping. It is felt that this data representation would require a highly nonlinear mapping, leading to extensive training time. Hence, this method was not pursued any further.

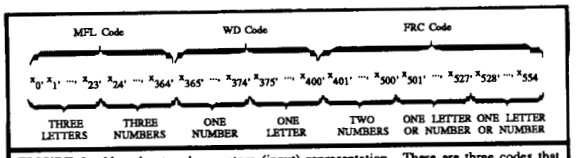


FIGURE 2: Neural network symptom (input) representation. There are three codes that make up each symptom description: MFL, WD, and FRC codes. Each code consists of numbers and/or letters that are transformed into a binary representation amenable to neural networks. The resulting input representation is a 555 element vector of binary values.

The third representation scheme (value-unit representation) assigns every possible input or output a separate PE. This method would produce the least difficult mapping because the high dimensional pattern space is the closest to being linearly separable. This representation, however, would have unacceptably slow computation times because the vectors are so large. In addition, such an approach would require enumeration of all possible inputs or outputs, a task beyond the scope of this effort. Because of the computation and data analysis requirements of this approach, encoding it is infeasible for our purposes.

The second representation scheme (intermediate-unit representation) represents the symptoms and actions taken with an encoding somewhere between the first and third schemes. We chose this method for the input because it represents a compromise between exhaustive data analysis and extensive computation time. The mapping is assumed to be linearly inseparable, yet computationally achievable within the projects time constraints

2.3.2. Input and Output Representations

Figure 2 shows how we represented the symptom (input) as a 555-element binary-valued vector. Because the input representation is well understood and static, the transformation from data base entry to binary vector was the only preprocessing necessary.

The fault (output) representation was more difficult. Because both single and multiple faults can be associated with the same symptom (input), the output representation has to maintain a running set of statistics that describe the likelihood of each possible fault given a particular input. In addition, the number of faults is not known apriori, hence it is necessary to develop a dynamic output representation that can grow with the number of faults being encoded in the network. To handle each of these issue,

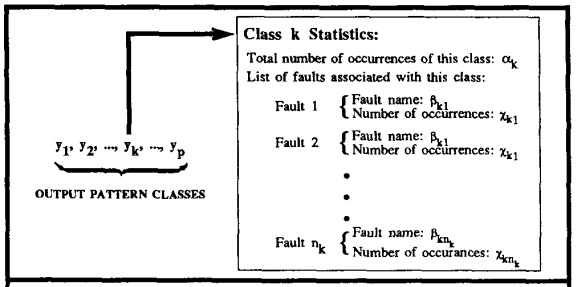


Figure 3: The output vector (fault) representation. Associated with each component of the output vector is a set of statistics kept in a data structure. Statistics kept include: (1) total number of occurrences of this particular output class, (2) the number of faults associated with this class, and (3) the number of times each individual fault within this class had occurred.

we encoded the output data in a dynamically sized 3 dimensional array that tabulates all faults that occur. **Figure 3** illustrates what information is stored with each pattern class processing element in the network and **Figure 4** shows the overall picture of the input and output representation with the neural network.

3. NEURAL NETWORK IMPLEMENTATION

3.1. WHICH NEURAL NETWORK IS BEST?

Choosing the proper neural network is essential to the success of this experiment. To determine which network is best suited to perform our pattern classification task requires a complete enumeration of the qualities desired in the overall system. These qualities include:

- ability to immediately classify symptoms;
- ability to handle noisy and incomplete data;
- ability to not favor heavily occurring symptoms over rarely occurring symptoms;
- ability to recognize novel symptoms;
- ability to immediately incorporate new symptoms; and
- ability to develop new symptom classes.

Originally we had intended to use the backpropagation [4] network, but quickly realized that its inability to detect sufficiently novel symptoms and its inability to immediately incorporate new symptom information were too limiting for its use. Another neural paradigm that we considered was the Learning Vector Quantization network [5] and its counterpropagation extension [6]. Although these networks do provide an ability to determine the novelty of a symptom, they suffered from an inability to immediately incorporate new symptoms and are not useful in this application. What was needed was a network that would allow new patterns to be immediately incorporated (i.e. on-line learning) and could distinguish between novel and known patterns. The answer is the binary adaptive resonance theory network (ART1) [7].

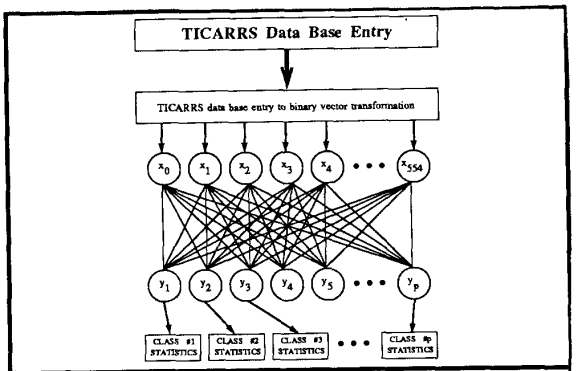


Figure 4: The overall view of the neural network representation scheme for diagnostics. The TICARRS data base entry is transformed into a 555-element input that represents that symptom. During both learning and operation the symptom is classified and each class has a set of dynamically changing statistics as well as a list of all the faults associated with the class.

3.2. BINARY ADAPTIVE RESONANCE THEORY (ART1)

The binary adaptive resonance theory (ART1) ANS — introduced by Carpenter and Grossberg [7] — is a two layer, nearest-neighbor classifier that stores an arbitrary number of binary spatial patterns $A_k = (a_{k1}, \ldots, a_{kn})$, k = 1, 2, ..., m, using a fast-learning version of the competitive learning law. ART1 learns on-line, operates in discrete or continuous time, and has the topology shown in **Figure 5**, where the n F_X PEs correspond to A_k's components and the p F_Y PEs each represent a pattern class. The connections from the F_X PEs to each F_Y PEs w_{ij} form reference vectors. For example, the connections from the F_X PEs to the F_Y PE b_j form the reference (weight) vector $W_j = (w_{1j}, w_{2j}, \ldots, w_{nj})$. The connections from the F_Y PEs to the F_X PEs form the pattern vectors. For example, the pattern vector attached to the *jth* F_Y PE is $V_j = (v_{j1}, v_{j2}, \ldots, v_{jn})$.

Although ART1 is principally a two layer architecture, it is also transparently framed within two subsystems — the attentional subsystem and the orienting

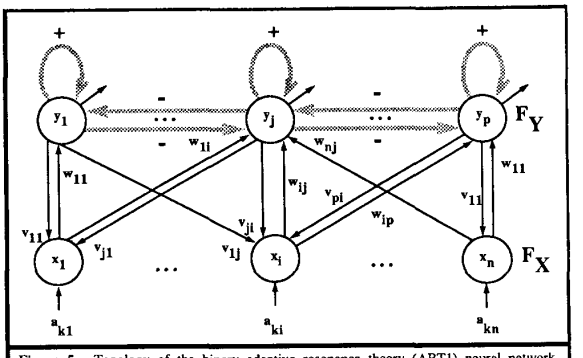


Figure 5: Topology of the binary adaptive resonance theory (ART1) neural network. There are inter-layer feedback connections (w_{ij} and v_{ji}) between the F_X and F_Y PEs that facilitate a resonation upon proper match between encoded patterns $V_j = (v_{j1}, v_{j2}, ..., v_{jn})$ and input patterns $A_k = (a_{k1}, a_{k2}, ..., a_{kn})$. The F_X PEs accept inputs from the environment and the F_Y PEs each represent a pattern class. During operation the F_Y PEs employ an invisible on-center/off-surround competition that is used to choose the proper class for the presented input. These lateral interactions are shown in the figure as shaded self-exciting (+) and neighbor-inhibiting (-) connections to emphasize this point. To keep the presentation uncluttered, all connections are not shown — there is actually a connection from each F_X PE to each F_Y PE and vice-versa, a shaded negative lateral connection from each F_Y PE to every other F_Y PE, and a shaded positive recurrent connection from every F_Y PE to itself.

subsystem. The attentional subsystem only allows the F_X PEs to be engaged when an input pattern is present. The orienting subsystem removes F_Y PEs from the set of allowable winners when an insufficient memory-input match occurs — an operation termed short-term memory (STM) reset. These subsystems are implemented as inherent operations during pattern processing.

In essence, ART1 conducts a serial search through the encoded patterns associated with each class trying to find a sufficiently close match with the input pattern. If no class exists, a new class is made. The test for a sufficient match between the top-down feedback pattern and the bottom-up input pattern is called hypothesis testing. An operational overview is provided in **Appendix One** and the equations are provided in **Appendix Two.**

3.3. PROBABILITY POST-PROCESSING

By tracking the number of occurrences of each class in the ART1 network (see **Figure 3**) and by tracking the number of occurrences of each fault that occurred within each class we are able to provide a frequency-based probability measure for each fault within a given class. In addition, as the diagnostic system is used and the number of fault occurrences increases, the statistics become more reliable.

As an example, assume that we are operating the system and we have just classified a symptom as belonging to class 123 — y_{123}. The hypothetical statistics associated with y_{123} are shown in **Figure 6.** This class has only three faults associated with it: (1) FCC FAILED, (2) INS NEEDS SERVICE, and (3) INS FAILED. The total

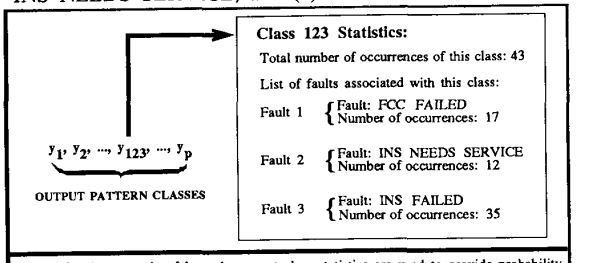


Figure 6: An example of how the output class statistics are used to provide probability information concerning which fault is associated with each symptom. The statistics show that this class has occurred 43 times. Of the 43 occurrences of this class, 17 times fault 1 occurred, 12 times fault 2 occurred and 35 times fault 3 occurred. Using this information, the probabilities of each fault being associated with the given symptom are 39.5%, 28.6%, and 81.4%, respectively

number of times that this class has occurred is 43. Each of the three faults occurred 17, 12, and 35 times respectively. To calculate the probability of each fault simply requires dividing the number of occurrences for each individual fault by the total number of occurrences of the class. For example, the probability that fault 1 is associated with the symptom is 17/43 = 0.395, or 39.5%.

4. CONCLUSION

We have developed an extension of ART1 that is used to implement an on-line learning multiple-fault diagnostic system that improves its performance with use. ART1 was found to be the most appropriate neural network for this application because it is able to quickly incorporate new patterns, it can inherently handle noisy and missing data, and it provides immediate responses. Also, with the addition of the output pattern statistics, we can overcome database ambiguities — a very difficult problem in classical AI systems.

ART1 is targeted for eventual use in diagnostic systems such as the Air Force Human Resources Lab's Model-based Diagnostic Aiding System (MDAS). The potential for this system is not strictly limited to this system. Because of the flexibility in construction and use, this system is easily applied to any weapon platform (as a generic diagnostic engine). When addressing the issue of flexibility, it is important to remember that neural networks can easily fuze data from several sensors — an ominous task for most diagnostic systems.

Information fusion comes easily because the network learns to map input vectors to pattern classes irrespective of the actual information contained within the input data. As an example, acoustic stress information and infrared images could be combined as inputs to a heli-rotor diagnostics system.

In this paper we have described the system that has been implemented to perform fault diagnosis of F-16 fighters on the flight line. In part II of this paper [8] we will discuss the results of our simulation experiments with actual TICARRS data and we will outline some areas of future exploration.

ACKNOWLEDGMENTS

We would like to thank J. Harold McBeth and William Bertch for their insightful comments and helpful critiques of this work. This work was supported by Air Force Human Resources Lab - Combat Logistics Branch contract number F33615-87-G0015 (IMIS-DD Program, Dave Gunning - POC).

REFERENCES

[1] Simpson, P., Artificial Neural Systems: Foundations, Paradigms, Applications and Implementations, Pergamon Press: Elmsford, NY, expected late 1989 release.

[2] TICARRS User's Manual Organizational and Intermediate Levels: Volume I and II, Air Force Logistics Command, LOC/TLPO, Wright Patterson Air Force Base, OH, 45433.

[3] Walters, D., Response mapping functions: Classification and analysis of connectionist representations, Proceedings of the IEEE First International Conference on Neural Networks: Volume III, pp. 75-86, SOS Press, San Diego, CA, 1987.

[4] Werbos, P., Beyond Regression: New tools for prediction and analysis in the behavioral sciences, Harvard University, Ph.D. Thesis, 1974.

[5] Kohonen, T., Self-Organization and Associative Memory, Springer-Verlag: Berlin, 1984.

[6] Hecht-Nielsen, R., Counterpropagation networks, Proceedings of the IEEE First International Conference on Neural Networks, Volume II, pp. 19-32, 1987.

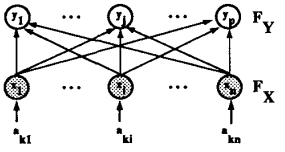
[7] Carpenter, G. and Grossberg, S., A massively parallel architecture for a self-organizing neural pattern recognition machine, Computer Vision, Graphics, and Image Understanding, 37, 54-115, 1987.

[8] McDuff, R. & Simpson, P. (1989). An investigation of neural networks for F-16 fault diagnosis: II. Simulation experiments (in preparation).

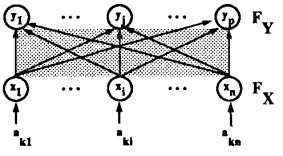
† General Dynamics Electronics Division, P.O. Box 85310, MZ 7202-K, San Diego, CA 92138
‡ Air Force Human Resources Labs, Combat Logistics Branch, Wright-Patterson AFB, OH 45433

APPENDIX ONE:
BINARY ADAPTIVE RESONANCE THEORY — OPERATIONAL OVERVIEW

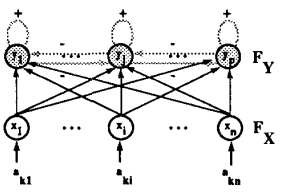
(1) Present an input pattern $A_k = (a_{k1}, ..., a_{kn})$ to the F_X PEs, where $F_X = (x_1, ..., x_n)$. Activity at the F_X layer is denoted by darkened PEs.



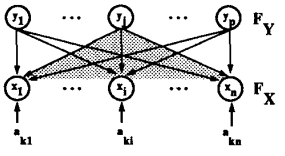
(2) Each F_X PE activation a_i sends a signal through its long-term memory (LTM) connections w_{ij} to every F_Y PE b_j. This flow of activation is denoted by the darkened regions flowing from F_X to F_Y.



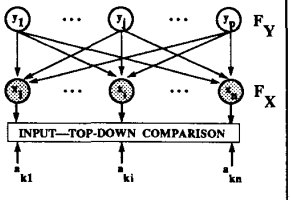
(3) Each F_Y PE competes with the others using competitive interactions until only one F_Y PE remains active. This operation, in essence, amounts to finding the reference vector W_j that most closely matches the input pattern A_k. This activity is denoted by the darkened F_Y PEs.



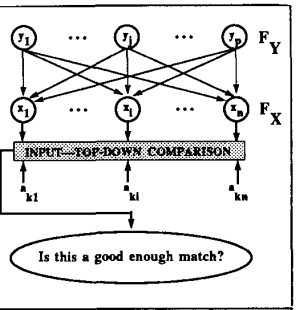
(4) The winning F_Y PE, say y_j, sends a top-down signal through its LTM connections, v_{ji}, back to F_X and (possibly) creates a new set of F_X PE activations. This activity is represented by the darkened region emanating from y_j.

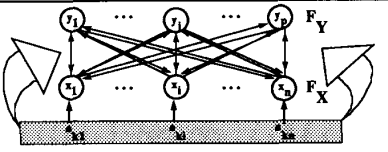


(5) The input, $A_k = (a_{k1}, ..., a_{kn})$ is compared with the top-down — F_Y to F_X — feedback activations $X = (x_1, ..., x_n)$ created by the winning F_B PE and the difference (Hamming distance) is measured between A_k and X. The F_X activity is represented by the darkened PEs and the comparison operation is denoted by the box.



(6) If the difference between X and A_k exceeds a predetermined value, the vigilance parameter, then the winning F_Y PE does not represent the proper class for A_k and it is removed from the set of allowable F_Y winners. If there are still F_Y PEs remaining in the set of allowable winners, go to step (2), otherwise recruit an uncommitted F_Y PE and encode A_k onto this PE's reference vector.





ENCODE PATTERN ONTO WEIGHTS

(7) If the difference between the two activations does not exceed the vigilance parameter, then y_j is represents the proper class for the input pattern A_k, and the input pattern is then merged with the stored pattern.

APPENDIX TWO:
BINARY ADAPTIVE RESONANCE THEORY — EQUATIONS

SYSTEM VARIABLES (Note — all vectors are binary valued):

$V_J = (v_{J1}, v_{J2}, ..., v_{Jn})$ = top-down pattern (template) recalled from the winning F_Y PE y_J.

$$X = (x_1, x_2, ..., x_n) = \begin{cases} A_k & \text{during steps (1)-(3)} \\ A_k \cap V_J & \text{during steps (4)-(6)} \end{cases}$$

$L > 1$

$\max\{1,D\} < B < 1+D$

$A, C \geq 0$

INITIALIZATION:

Direct Access Inequality:

$$0 < w_{ij} < L / (L-1+n)$$

Template Learning Inequality:

$$(B-1)/D < v_{ji} \leq 1.$$

F_Y CLASS SEARCH:

$$y_j = \begin{cases} 1 & \text{if } T_j = \max\{T_k: k \in J\} \\ 0 & \text{otherwise} \end{cases}$$

$$T_k = D \sum_{i=1}^{n} x_i w_{ik}$$

J = index set of allowable F_Y PE winners

F_Y PE RESET RULE:

Remove the winning F_Y PE y_J from the set of allowable winners if:

$$\frac{|A_k \cap V_J|}{A_k} < \rho \qquad |X| = \sum_{i=1}^{n} x_i$$

where $0 < \rho \leq 1$ is the vigilance paramter. The closer ρ is to 1, the finer the classification granularity.

LEARNING LAWS (FAST LEARNING):

Bottom-Up Learning: Only change the connections that abut the winning PE y_J:

$$w_{iJ} = a_{ki}$$

Top-Down Larning: Only change the connections emanating from the winning PE y_J:

$$v_{Ji} = L/(L-1+|X|)$$