

NAVAL HEALTH RESEARCH CENTER

DYNAMIC AGGREGATION OF SYMPTOM DATA

*J. E. Angus
R. Williamson
D. Pick
K. Ames
A. Niknejad
A. Ornatsky
X. Wu*

Report No. 95-34

19960401 165

INTERIM REPORT

Approved for public release: distribution unlimited.



NAVAL HEALTH RESEARCH CENTER
P. O. BOX 85122
SAN DIEGO, CALIFORNIA 92186 - 5122

NAVAL MEDICAL RESEARCH AND DEVELOPMENT COMMAND
BETHESDA, MARYLAND



DYNAMIC AGGREGATION OF SYMPTOM DATA

Prepared for:

NAVAL HEALTH RESEARCH CENTER
P.O. Box 85122
San Diego, CA 92186-5122

Prepared by:

The Claremont Graduate School Mathematics Clinic
Professor John E. Angus, Director
Professor Robert Williamson¹
Mr. Daniel Pick²
Kevin Ames³
Amir Niknejad³
Alex Ornatsky³
Xuemei Wu³

¹ Faculty Consultant

² Team Leader

³ Team Members

Report 95-34 Supported by the Navy Medical Research and Development Command under work unit number 63706N-M0095.005-6512. The views expressed in this report are those of the authors and do not reflect the official policy or position of the Department of the Navy, Department of Defense, or the U.S. Government. Approved for public release, distribution unlimited.

Summary

Problem: Allow epidemiologists to examine ways to obtain and assess clinical data as quickly and accurately as possible while in a combat setting. Epidemiologists and preventive medicine specialists need ways to relate patient signs and symptoms data collected at the time of presentation into associated categories to help in the early detection of any adverse health event.

Objective: To develop a computerized algorithm that would accept sequentially in real time patient symptom data (vectors) and dynamically identify patterns of probable syndromes or diseases so timely preventive measures can be taken.

Approach: A simple Adaptive Resonance Theory (ART) algorithm was developed that was capable of sequentially accepting patient symptom vectors and dynamically clustering them into patterns or syndromes described by prototype symptom vectors. A modified ART algorithm was also proposed to deal with syndromes that do not possess a strong core of symptoms.

Results: The basic ART algorithm was demonstrated with simulated symptom data from 300 simulated patients representing 10 underlying syndromes. The algorithm was shown to correctly identify the syndromes and to cluster the patients into their correct syndromes.

Conclusions: Based on the results of this simulation, ART algorithms appear to accomplish the stated objective for simulated data. The basic ART algorithm is simple, easily implemented, and capable of running in real time on modest computing resources (e.g., a laptop PC). Future work on these algorithms should include testing on real field data and experimentation to determine optimum settings for the algorithm parameters.

1. INTRODUCTION	4
2. STUDY OF APPROACHES	5
3. DYNAMIC AGGREGATION ALGORITHM	7
4. EXPERIMENTAL RESULTS	9
5. ALTERNATE ART ALGORITHM	12
6. FUTURE WORK	15
REFERENCES	16
APPENDIX I. QBASIC CODE FOR THE ART ALGORITHM	18
APPENDIX II. QBASIC SYMPTOM SIMULATOR	25
APPENDIX III. 300 SIMULATED SYMPTOM VECTORS	27
APPENDIX IV. APPLICATION OF THE ART ALGORITHM TO SIMULATED SYMPTOM DATA	33
APPENDIX V. PROTOTYPE GRAPHICAL USER INTERFACE	40

1. INTRODUCTION

The health and well-being of groups of U.S. service personnel assigned to duty in foreign countries is of vital importance to the successful completion of the intended missions. Upon relocation, such groups represent nonindigenous populations exposed to the endemic diseases of the region and other hazards associated with the assignment. Within these groups, outbreaks of debilitating diseases/syndromes can render many individuals unable to perform their duties, placing a severe strain on medical resources, jeopardizing their missions, and possibly endangering the lives of those stricken.

Field medical personnel need a way to continuously monitor the population and to quickly detect patterns of affliction. Since the number of medical doctors available in the field is usually very limited, this is of particular importance to corpsman who do not have the training or resources to make a definitive diagnosis in the field. Typically, a sick patient is examined in the field by a corpsman. The corpsman records signs and symptoms for the patient and enters them into a database. The definitive diagnosis and etiology of the disease/syndrome, however, are not determined until much later when the patient is transferred to a field hospital or back to the home base where doctors can perform diagnostics and laboratory analyses. One consequence of this delay is that as patients enter this system over time, clusters of patients afflicted with the same disease/syndrome can form and go undetected until there is a high toll.

What is needed is a method of examining the sign-symptom profiles in near real time, as well as detecting clusters of affliction, which might be called "syndromes," and their associated descriptions so that preventive measures can be instituted before the loss of productive personnel can drain resources. For example, a cluster of patients with severe gastroenteritis will lead authorities to examine food, water, and sanitary conditions first as obvious culprits in an attempt to locate the source and limit the number of new cases. Note that the emphasis here is on timely detection and prevention; the first priority in the field is to stop the progression rather than a definitive diagnosis of the disease or determination of its etiology.

2. STUDY OF APPROACHES

The problem of aggregating sign-symptom data falls into the general theme of classification, clustering, and pattern recognition. There is a plethora of medical research in which classification, clustering, and pattern recognition methods have been useful.* Two examples are the work of Ciampi et.al.,⁴ and Kristensen.¹¹ Unfortunately, the problem of interest in this study does not fit neatly into any of the classical paradigms for which analytical solutions are available (See, for example, Nadler and Smith,¹⁷ and Fukunaga⁷ for mathematical descriptions of these classical paradigms). Generally, algorithms for classification, clustering, and pattern recognition involve either supervised or unsupervised learning, and the number of clusters present is assumed known in advance.

In supervised learning, a large representative training sample of individual sign-symptom vectors with associated cluster information (e.g., definitive diagnosis) are used to "train" a complex algorithm to recognize the patterns associated with each cluster. Here, "training" usually entails repeatedly presenting the training data to the algorithm while continuously adjusting a number of parameters until a desired classification accuracy is achieved. Variations on this include the use of cross-validation (holding certain individuals out of the training sample for measuring accuracy). Computerized disease diagnosis algorithms are available and in use today, some of which may be purchased at local software stores and run on personal computers equipped with CD ROM (compact disk, read only memory). Most conduct an interview with the patient (or attending medical person), asking questions until a certainty threshold for diagnosis is exceeded. The success of these systems is completely determined by the reliability, accuracy, and size of the training database.

* A comprehensive bibliography on pattern recognition and cluster analysis is included in the reference section of this report. For more information, please see references 1-9, 12, 13-17, and 21.

A variation on supervised learning makes use of expert opinion in place of data. Intriguing work in this area includes that of Saaty,^{18, 19} who developed a mathematical technique for making decisions based on inconsistent expert opinions. In the problem of interest to this study, usually insufficient or no training data are available. This is true because the symptomology associated with endemic diseases may be well understood for an indigenous population, but not so with a nonindigenous population. Also, the unique circumstances associated with any military mission can lead to unprecedented epidemics and conditions. These considerations rule out the use of supervised learning.

In unsupervised learning, the algorithm is designed to form clusters based on patterns existing in a large set of data that have been collected. Unlike supervised learning, the data provided include the patterns but none contain the true classification or cluster membership information. One successful algorithm is the "K-means" algorithm as described in Schalkoff.²⁰ Implicit in this and other algorithms that use unsupervised learning is the assumption that K (known) clusters exist. In the problem addressed by this study, the number of clusters is unknown in advance, and the data become available sequentially (as opposed to being supplied in one large complete group). It is possible to guess an upper bound on the number of clusters. For example, one could take the number of incapacitating endemic diseases for the region and add one or two to allow for the possibility of the emergence of a new disease or syndrome. Still, there is no way to know which, and how many, clusters will become active until data are entered.

3. DYNAMIC AGGREGATION ALGORITHM

What is needed for the problem faced in this investigation is a real-time unsupervised learning system than can adapt to the sign-symptom data as they are collected. Adaptive Resonance Theory (ART) gives several prototypes of such a system (See Hertz et.al.¹⁰ for a discussion and further references to ART). We have finished coding an ART algorithm (later referred to as "ART1") that seems to satisfy the need for the algorithm to adapt. To describe it, some assumptions and special notations are needed. First, it is assumed that each patient can be represented by a binary vector of fixed length n . Each component of the symptom vector represents a given sign or symptom, with the presence or absence of that sign or symptom indicated by a 1 or 0, respectively. Next, we assume that no more than K classes or clusters exist. This value need not be exact, but care should be taken to ensure that it is at least an upper bound on the total number of possible clusters (e.g., K = the number of debilitating endemic diseases + 5). Associated with each value of the index $i = 1, 2, 3, \dots, K$ is a prototype vector w_i . Initially, $w_i = \underline{1} = (1, 1, \dots, 1)$ for each i , that is, an n -vector of 1s. For vectors a and b , the notation $a \bullet b$ signifies the usual vector dot product. For binary vectors u and v , $u(\text{AND})v$ denotes the binary vector formed by multiplication of the respective components of u and v . The number ε is a small positive number, for example, $\varepsilon = 0.01$. Finally, ρ , $0 < \rho < 1$, is a "tuning parameter" that is set in advance. As will be seen later, it is a "vigilance" parameter, controlling the ease with which new clusters come into existence, as detected by the algorithm. A generic sign-symptom vector for a patient is denoted by ξ . The ART algorithm is described in steps 0 through 5 as follows.

0. If there are no more data, stop. Otherwise, present the next symptom vector ξ , and classify it as "unclassified."
1. "Enable" all clusters, $i = 1, 2, \dots, K$.
2. Determine i^* such that

$$\left(\frac{\xi \bullet w_{i^*}}{w_{i^*} \bullet \underline{1} + \varepsilon} \right) = \max \left\{ \frac{\xi \bullet w_i}{w_i \bullet \underline{1} + \varepsilon} : i \text{ corresponds to an enabled cluster} \right\}$$

3. Compute

$$r = \frac{w_{i^*} \bullet \xi}{\xi \bullet \underline{1}}$$

If $r < \rho$ then “disable” cluster i^* and go to Step 2. If there are no enabled clusters left, then leave ξ unclassified and go to Step 0. Otherwise, go to Step 4.

4. Classify ξ into cluster i^* , and adjust w_{i^*} by replacing it with $w_{i^*}(\text{AND})\xi$.
5. Go to Step 0.

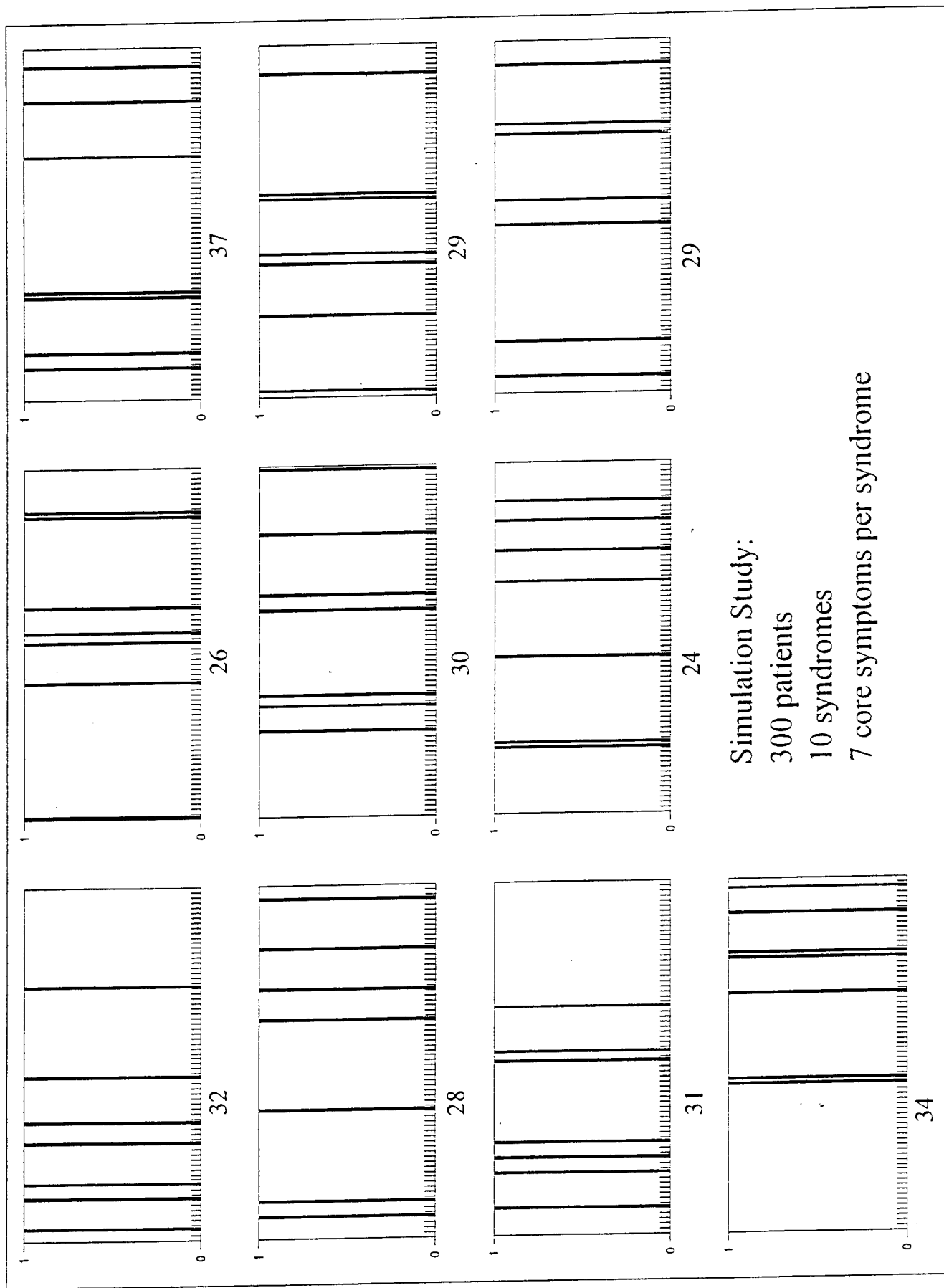
This algorithm solves the various problems raised earlier with more conventional clustering algorithms. It forms new clusters as needed (up to a maximum of K) and produces a prototype vector that represents each cluster (i.e., the w -vectors). The significance of the vigilance parameter ρ is also clear. As ρ becomes closer to 1, higher correlation with an existing prototype is required in order to classify a vector into that cluster, and hence it is more likely that a new cluster will be established. Conversely, as ρ becomes closer to 0, less correlation is needed to classify a vector into an existing cluster, and hence new clusters are less likely to be formed.

4. EXPERIMENTAL RESULTS

This algorithm has been coded in Microsoft Qbasic (source listing in Appendix I). A prototype with graphical user interface for demonstration purposes was also coded in Microsoft Visual Basic (see Appendix V for a picture of that interface). Both versions were tested on simulated symptom vectors with a high degree of success. We created a simulated database of 300 patient symptom vectors using Qbasic code shown in Appendix II. This code forms 300 symptom vectors of length 70 using the following method. First, 10 core syndrome vectors are formed by selecting 7 symptoms for each (symptoms do not overlap). A core symptom vector is chosen at random, and the current symptom vector is set equal to it. Independently, each symptom not present is added with probability $2/63$. (Thus, an average of 2 symptoms are added over and above the core symptoms). The symptom vector is then output. This process is repeated 300 times. One run yielded the data shown in Appendix III. Because patients are randomly selected to have one of the 10 syndromes, the expected number of patients (out of 300) representing each syndrome is 30. For the data in Appendix III, there were 30, 34, 29, 32, 24, 26, 31, 37, 28 and 29 representatives from the syndromes 1,2,...,10, respectively.

The ART algorithm was presented with these data and, processing each patient sequentially, was able to correctly identify all 10 syndromes and correctly classify each of the 300 patients into the correct syndrome. The result of running the ART algorithm is shown in Appendix IV (ART assigns different numerical names to the syndromes than those originally assigned). A visual display of the 10 syndromes detected by the ART algorithm is shown in the Figure 1, where each syndrome cluster prototype is represented by a bar graph with each symptom present depicted by a black bar. The number below each prototype is the number of patients aggregated into that cluster by the ART algorithm. Note that each prototype has 7 symptoms (the number of core symptoms originally specified in the simulation). These are exactly the original core symptom vectors used to generate the input data to the ART algorithm. Note that ART does not

Figure 1. Syndrome Prototypes Detected by the Art Algorithm



provide a diagnosis for each cluster, only a prototype symptom vector. It will be the job of physicians and epidemiologists to deduce from the symptoms in the prototype vector the probable etiology and/or preventive action to be taken.

5. ALTERNATE ART ALGORITHM

In one scenario the ART algorithm will perform in a way that may not be appropriate. That scenario is one in which the manifestations of the prototype diseases are highly variable. Notice that in Step 4 of the ART algorithm, the prototype vector representing a cluster is "masked" by each new vector that is classified into that cluster. Suppose, for example, that after a large number of vectors have been clustered, the prototype vector for disease cluster 1 is (000111000). If the vectors (000011000), (000001000) are then presented, in that order, it would be possible for all of these to be classified into this same cluster. If this happens, the prototype vector would degenerate into (000001000), which is no longer an adequate descriptor of the cluster. This situation would be caused by high variation in how a disease affects the individuals in the population. It would not occur if each syndrome has its own strong core of symptoms (i.e., symptoms that are always present for patients afflicted with that syndrome). Whether this can be a problem with this ART algorithm will depend on the analysis of some real data, and whether this is desirable must be determined by the user. We have developed an alternate ART algorithm that is more robust to the situation described here. The choice of classifying algorithm depends on the answer to several questions, including these:

1. Must the classes be described by prototypes whose components are 0 or 1 only?
2. Should a particular symptom be removed from a syndrome prototype if even one patient who otherwise displays the symptoms of the syndrome does not display the particular symptom in question (i.e., should syndromes be masked?)?
3. Should past information be dominated by more recent information? Or should old observations and new ones be weighted equally?
4. What should be the measure of closeness of symptom vectors?

For certain problems, the answer to question 1 is “yes,” and the ART1 algorithm is suitable. In that case the answer to question 2 is almost forced to be “yes,” also. If the answer to the first two questions is “no,” then a very simple alternate scheme is available, which we call the “centroid algorithm.” Two parameters in that algorithm are set to reflect the answer to the third question: all patients may weigh equally, or the weight laid on old observations may be allowed to diminish. The effectiveness is relatively insensitive to the choice of measure of closeness, although it is sensible to give some thought to this point. Here, we use the cosine of the angle between the symptom vectors. This says that the difference measure between them is related to the ratio of the number of symptoms on which they disagree to the number of symptoms each presents. Other measures might well be used (e.g., just the number of symptoms on which they differ; of course then the vigilance parameter would not be between 0 and 1).

In brief, the algorithm is: first, initialize all prototypes $w_i = 0$, then read in a symptom vector s and find the i that maximizes, over all classes in use, the cosine of the angle between w_i and s , that is:

$$\frac{w_i \bullet s}{\|w_i\| \cdot \|s\|}$$

Here, $\|*\|$ is the usual Euclidean norm. If that value exceeds the vigilance parameter, then assign syndrome i as the class of the patient. If not, then put a new class in use, and define its syndrome prototype vector to be s , unless all possible classes have been used. If the classification was successful, then update the prototype syndrome vector according to:

$$w_i \leftarrow a \cdot w_i + b \cdot s$$

where a , b represent the weight of old and new observations: $a = 1$, $b = 1$ corresponds to equal weight, and convex combinations $a + b = 1$ correspond to eroding the weight of past observations; the smaller the value of a relative to b , then the faster the rate at which past observations are eroded. This algorithm yields results similar to those of

ART1 on our data, and does not mask the classes as ART1 does. The components of the syndrome will represent values between 0 and 1. Those symptoms that occur more often will have values close to 1. In general, the most frequent symptoms will have higher values than those that appear less frequently. If a and b are set to 1, then after we add all symptom vectors to the syndrome and divide each element of the syndrome vector by the number of symptom vectors assigned to it, we will get a vector where each component represents a probability of occurrence of each particular symptom. Preliminary results indicate that this algorithm is also very effective in classifying data of the type generated to test ART1, for which masking is appropriate, but it is also effective with data that present "missing symptoms."

6. FUTURE WORK

The experimental work with the ART algorithm should be continued. The next step regarding the experimentation will be to develop procedures or guidelines for setting the vigilance parameter. This can be accomplished with further analysis and simulation studies. Another important activity is testing the Alternate ART algorithm, in particular, the influence of the values of the a and b parameters on the rate of erosion of previous observations. For both algorithms, it is important to perform tests on real field data.

REFERENCES

1. Amari, S. and Takeuchi, A. (1978). *Mathematical theory on formation of category detecting nerve cells* (vol. 29, pp. 127-136). Biological Cybernetics.
2. Beni, G. and Liu, X. (1994). *A least biased fuzzy clustering method* (vol. 16, no. 9, pp. 954-960). IEEE Transactions on Pattern Analysis And Machine Intelligence.
3. Bock, P. (1993). *The emergence of artificial cognition: an introduction to collective learning* (pp. 137-184). Singapore: World Scientific.
4. Ciampi, A., Schiffrin, A., Thiffault, J., Quintal, H., Weitzner, G., Poussier, P. and Lalla, D. (1990). *Cluster analysis of an insulin-dependent diabetic cohort towards the definition of clinical subtypes* (vol. 43, pp. 701-715). Journal of Clinical Epidemiology.
5. Cutler, A. and Breiman, L. (1994). *Archetypal analysis* (vol. 36, no. 4, pp. 338-347). Technometrics.
6. Fisher, D. and Langley, P. (1988). *Conceptual clustering and its relation to numerical taxonomy* (pp. 77-116). In: W. A. Gale, ed., *Artificial intelligence and statistics*. Mass: Addison - Wesley.
7. Fukunaga, K. (1990). *Statistical pattern recognition*. New York: John Wiley.
8. Ganter, B. and Wille (1989). *Conceptual scaling* (pp. 139-167). In: F. Roberts, ed., *Application of combinatorics and graph theory to the biological and social sciences*. New York: Springer - Verlag.
9. Grossberg, S. (1976). *Adaptive pattern classification and universal recoding: part I. Parallel development and coding of neural feature detectors* (vol. 23, pp. 121-134). Biological Cybernetics.
10. Hertz, J., Krogh, A. and Palmer, R. G. (1991). *Introduction to the theory of neural computation*. Redwood City, CA: Addison Wesley.
11. Kristensen, Petter (1992). *Bias from nondifferential but dependent misclassification of exposure and outcome* (vol. 3, pp.210-215). Epidemiology.
12. Kruskal, J. B. (1964). *Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis* (vol. 29, no. 1, pp. 1-27). Psychometrika.

13. Krzanowski, W. J. and Lai, Y. T. (1988). *A criterion for determining the number of groups in a data set using sum - of - squares clustering* (vol. 44, pp. 23-34). Biometrics.
14. Marriott, F. H. C. (1971). *Practical problems in a method of cluster analysis* (pp. 501-514). Biometrics.
15. Milligan, G. W. (1985). *An examination of procedures for determining the number of clusters in a data set* (vol. 50, pp. 159-179). Psychometrika.
16. Milligan, G. W. (1980). An examination of the effect of six types of error perturbation on fifteen clustering algorithms (vol. 45, no. 3, pp. 325-342). Psychometrika.
17. Nadler, M. and Smith, E. (1993). *Pattern recognition engineering*. New York: John Wiley.
18. Saaty, T. L. (1977). *A scaling method for priorities in hierarchical structures* (vol. 15, pp. 234-281). Journal of Mathematical Psychology.
19. Saaty, T. L. (1991). A natural way to make momentous decisions (vol. 51, pp. 561-571). Journal of Scientific & Industrial Research.
20. Schalkoff, R. (1992). *Pattern recognition. Statistical, structural, and neural approaches*. New York: John Wiley.
21. Tenenhaus, M. (1985). *An analysis and synthesis of multiple correspondence analysis, optimal scaling, dual scaling, homogeneity analysis and other methods for quantifying categorical multivariate data* (vol. 50, no. 1, pp. 91-119). Psychometrika.
22. Wenzel, Richard P. (1986). *The evolving art and science of hospital epidemiology* (vol. 153, pp. 462-470). Journal of Infectious Diseases.

Appendix I. QBASIC Code for the ART Algorithm

```
REM This is a simple adaptive resonance algorithm.
REM Professor J. Angus - Mathematics Clinic - Spring Term, 1995
REM
REM W holds the prototype vectors
REM S holds the current data vector
REM VALUES holds the current "scores"
REM NORDER holds the indices of the sorted VALUES
REM rho is the threshold for determining a new cluster
REM k is the maximum number of clusters
REM sort takes arrayin and sorts it into arrayout in ascending order.
REM norder maintains the same permutation of 1, 2 ,..., n as was
REM determined by sorting arrayin.
REM wdot1 computes the dot product of the iith row of w with the vector
REM of all ones
REM eps is epsilon (value used to help break ties)
REM
DECLARE SUB sort (arrayin(), arrayout(), norder(), n)
DECLARE SUB wdot1 (ii!, sum!, n)
DIM SHARED w(20, 70)
DIM s(70), counts(20)
DIM norder(20), values(20)
k = 20
rho = .5
eps = .01
REM open input data file
OPEN "d:\nhrc95\symptom.txt" FOR INPUT AS #1
REM
REM open output data file
OPEN "d:\nhrc95\testout.txt" FOR OUTPUT AS #2
REM read the number of columns in the symptom vectors
INPUT #1, ns, ncases
REM
REM initialize the prototype vectors.
FOR i = 1 TO 20
    counts(i) = 0
    FOR j = 1 TO ns
        w(i, j) = 1
    NEXT j
NEXT i
FOR ica = 1 TO ncases
    LINE INPUT #1, data$
    actual = VAL(MID$(data$, 72))
    FOR i = 1 TO ns
        s(i) = VAL(MID$(data$, i, 1))
    NEXT i
    sdot1 = 0
REM
REM compute dot product of data vector with vector of ones
REM
```

```

    FOR i = 1 TO ns
        sdot1 = sdot1 + s(i)
    NEXT i
    REM initial class is 0="unclassified"
    iclass = 0
    FOR i = 1 TO 20
        values(i) = 0
        norder(i) = i
        CALL wdot1(i, sum, ns)
    REM
    REM compute the scores
    REM
        FOR j = 1 TO ns
            values(i) = values(i) + w(i, j) * s(j) / (sum + eps)
        NEXT j
    NEXT i
    FOR i = 1 TO 20
        'PRINT #2, values(i);
    NEXT i
    'Print #2, " "
    REM
    REM sort the scored and keep track of the original indices
    CALL sort(values(), values(), norder(), k)
    'FOR i = 1 TO 20
    'PRINT #2, values(i);
    'NEXT i
    'PRINT #2, " "
    'FOR i = 1 TO 20
    'PRINT #2, norder(i);
    'NEXT i
    'PRINT #2, " "
    REM
    REM Start with the best score, and check the correlation criterion.
    REM If a classification into an existing class can be made, jump out.
    REM If not, decrement k by 1 and try the next best.
    REM
        FOR ist = k TO 1 STEP -1
            istar = norder(ist)
            wdots = 0
            FOR i = 1 TO ns
                wdots = wdots + s(i) * w(istar, i)
            NEXT i
            IF wdots / sdot1 >= rho THEN
                'PRINT #2, istar, wdots / sdot1
                iclass = istar
                GOTO outofit
            END IF
        NEXT ist
    REM
    outofit:
    REM
    REM Update prototype vector, print out data vector and its class.
    REM
        FOR i = 1 TO ns

```

```

        w(iclass, i) = w(iclass, i) * s(i)
    NEXT i
    PRINT #2, MID$(data$, 1, 70) + " " + STR$(iclass) + " " + STR$(actual)
    counts(iclass) = counts(iclass) + 1
NEXT iclass
FOR j = 20 TO 1 STEP -1
    PRINT #2, counts(j)
NEXT j
total = 0
FOR j = 1 TO 20
    total = total + counts(j)
NEXT j
PRINT #2, total
FOR j = 1 TO 20
    proto$ = ""
    IF counts(j) > 0 THEN
        FOR i = 1 TO 70
            proto$ = proto$ + MID$(STR$(w(j, i)), 2, 1)
        NEXT i
        PRINT #2, proto$ + " " + STR$(counts(j))
    END IF
NEXT j
PRINT "Done"
END

```

```

SUB sort (arrayin(), arrayout(), norder(), n)
DIM ra(n)
STATIC ra
FOR i = 1 TO n
    ra(i) = arrayin(i)
NEXT i
begin:
switch = 0
FOR i = 2 TO n
    j = i - 1
    IF ra(j) > ra(i) THEN
        save = ra(j)
        ra(j) = ra(i)
        ra(i) = save
        iord = norder(j)
        norder(j) = norder(i)
        norder(i) = iord
        switch = 1
    END IF
NEXT i
IF switch = 1 GOTO begin
FOR i = 1 TO n
    arrayout(i) = ra(i)
NEXT i
END SUB

```

```

SUB wdot1 (ii, sum, n)
DIM i
sum = 0

```

```

FOR i = 1 TO n
    sum = sum + w(ii, i)
NEXT i
END SUB
REM *****
REM *
REM * Adaptive Resonance Theory Program for *
REM * Dynamic Clustering *
REM * CGS Mathematics Clinic Spring 1995 *
REM * Alex Ornatsky *
REM * Amir Niknejad *
REM *
REM *****
REM
COMMON SHARED n
DECLARE FUNCTION dot (vx(), vy())
DECLARE FUNCTION norm (vx())
CLS
PRINT "Input the # of clusters: ";
INPUT k
PRINT "Input the # of symptoms: ";
INPUT n

DIM w(1 TO k, 1 TO n)
DIM c(n)
DIM w1(k, n)
DIM x(n)
DIM y(n)
DIM en(k)
DIM u(k)
DIM h(n)
DIM w20(n)
DIM symp(n)
DIM w2(n)
DIM w3(n)
DIM w4(k, n)

PRINT
REM k is the number of clusters
u2 = 0

PRINT
REM n is the number of symptoms
FOR i = 1 TO k
    FOR j = 1 TO n
        w(i, j) = 1
    NEXT j
NEXT i
PRINT

REM w(i,j) is the initial prototype vector
FOR i = 1 TO k

    u(i) = 0

```

```

NEXT i
PRINT

REM en[1 to k] is enabling indicator
REM u[1 to k] is the indicator for number of vectors assigned to that
REM prototype
REM PRINT "Input vigilance parameter : ";
REM e-the threshold parameter
vigilance = .5
eps = .01

OPEN "a:\nhrc30.dat" FOR INPUT AS #1
DO WHILE (NOT EOF(1))

50  FOR i = 1 TO n
    REM  PRINT "Input Symptom"; i;

        INPUT #1, symp(i)
    REM  INPUT symp(i)
    NEXT i
    PRINT "symp vector is:"
    FOR i = 1 TO n
        PRINT symp(i);
    NEXT i
    PRINT

    FOR i = 1 TO k
        en(i) = 1
    NEXT i
    FOR i = 1 TO k
        IF en(i) = 1 THEN
            FOR j = 1 TO n
                h(j) = w(i, j)
                c(j) = 1
            NEXT j
            b = dot(h(), c())
            epsnorm = eps + b
            FOR j = 1 TO n
                w1(i, j) = w(i, j) / epsnorm
            NEXT j
            END IF
        NEXT i

100  FOR i = 1 TO k
        IF en(i) = 1 THEN
            FOR j = 1 TO n
                w20(j) = w1(i, j)
            NEXT j
            w2(i) = dot(w20(), symp())
        END IF
    NEXT i
    d = 0

```

```

f = 0
FOR i = 1 TO k
  IF en(i) = 1 THEN

    IF w2(i) >= d THEN
      d = w2(i)
      f = i
    END IF
  END IF
NEXT i
IF en(f) = 1 THEN
  FOR j = 1 TO n
    w3(j) = w(f, j)
  NEXT j

  R = dot(w3(), symp())
  s = 0
  FOR j = 1 TO n
    s = s + symp(j)
  NEXT j
  IF s > 0 THEN
    R = R / s
  END IF
END IF
IF R < vigilance THEN
  en(f) = 0
  sum1 = 0
  FOR i = 1 TO k
    sum1 = sum1 + en(i)
  NEXT i

  IF sum1 > 0 THEN
    GOTO 100
  ELSE GOTO 200
  END IF
END IF
IF R >= vigilance THEN
  u(f) = u(f) + 1

  done = 1
  ELSE done = 0
END IF
IF done = 1 THEN
  PRINT "winning index is: "; f,
  FOR j = 1 TO n
    w4(f, j) = w(f, j) * symp(j)
    PRINT w4(f, j);
  NEXT j
  FOR j = 1 TO n
    w(f, j) = w4(f, j)
  NEXT j
  PRINT " "
  GOTO 250
END IF

```

```

200 u2 = u2 + 1
    PRINT "ATTENTION : NO MATCH!!!"
    PRINT "NUMBER OF UNMATCHED VECTORS:";
    PRINT u2

250 PRINT "enabling vector is:";
    FOR i = 1 TO k
        PRINT en(i);
    NEXT i
    PRINT " "
    PRINT "# of vectors/cluster:";
    FOR i = 1 TO k
        PRINT u(i);
    NEXT i
    REM PRINT "Continue? YES-1, No-0";
    REM INPUT v
    REM IF v = 0 THEN
    REM     GOTO 300
    REM     ELSE GOTO 50
    REM END IF
    PRINT
    LINE INPUT "Press Return for the next symptom vector"; mystring$

LOOP
CLOSE #1

300 END

FUNCTION dot (x(), y())
sum = 0

FOR i = 1 TO n

    sum = sum + x(i) * y(i)

NEXT i
dot = sum
END FUNCTION

FUNCTION norm (a())
norm = dot(a(), a())
END FUNCTION

```

Appendix II. QBASIC Symptom Simulator

```
REM Symptom simulator.
REM CGS Mathematics Clinic, Professor John Angus, Spring Term, 1995
REM
DIM count(10), permute(70)
REM
REM Initialize the symptom vector.
REM
s$ = "0000000000000000000000000000000000000000000000000000000000000000"
per$ = s$
OPEN "d:\nhrc95\symptom.txt" FOR OUTPUT AS 1
PRINT #1, 70, 300
FOR i = 1 TO 10
    count(i) = 0
NEXT i
RANDOMIZE
REM
REM Generate a random permutation of the integers 1, 2, ..., 70
REM
FOR j = 1 TO 70
    permute(j) = j
NEXT j
FOR j = 70 TO 2 STEP -1
    i = INT(RND * j) + 1
    p = permute(j)
    permute(j) = permute(i)
    permute(i) = p
NEXT j
FOR j = 1 TO 70
    PRINT permute(j);
NEXT j
REM
REM Begin loop to create 300 patients.
REM
FOR i = 1 TO 300
    REM
    REM Select a syndrome at random.
    REM
    id = INT(10 * RND) + 1
    REM
    count(id) = count(id) + 1
    REM
    REM Insert the core symptoms into the current symptom vector.
    REM
    FOR j = 1 TO 70
        MID$(s$, j, 1) = "0"
        IF j <= 7 * id AND j >= 7 * (id - 1) + 1 THEN MID$(s$, j, 1) = "1"
    NEXT j
    REM
    REM Randomly add an average of 2 additional symptoms to the symptom vector.
    REM
    u = RND
    IF u <= 2 / 63 AND MID$(s$, j, 1) = "0" THEN MID$(s$, j, 1) = "1"
```

```

REM
    NEXT j
REM
REM    Apply the same random permutation to each symptom vector.
REM
    FOR j = 1 TO 70
        t$ = MID$(s$, permute(j), 1)
        MID$(per$, j, 1) = t$
    NEXT j
REM
REM    Print the symptom vector.
REM
    PRINT #1, per$ + " " + STR$(id)
NEXT i
REM
REM    Print out the number of patients in each syndrome cluster.
REM
    FOR j = 1 TO 10
        PRINT #1, count(j)
    NEXT j
END

```

Appendix III. 300 Simulated Symptom Vectors

[illegible]

[illegible]

[illegible]

001000001001000000010001001010001100000000000000010000000000000000 2
1000000000000000010100000000100000001010100100000000000000001100000000 3
00000010010000000000011000000000000000000000000000000000000000000000 5
01001001000000000000000001100000101000000000100000100000001010010000100 7
000000000000000110000000000000000100000000100000100000110000100010000000 8
0010000010010000000100010000000001000000000000000100010000000000000000 2
000010010000000000000000000100000000000000001000001000000001001000000100 7
0000000000000001100000000000000001000000000000000100000100000100010010000 8
100000010000000000001000000001000000010100001000000000000001000001100000010 3
00010100001000000000010000100010010000100000000000001011000000000010000 6
00100000100100000001000100000000100000000000000000000000000000000000 2
00001001000000000000000000010000000000000000000000000000000000000000 7
000000000001000001000010100000001000000000100100000000000100000000001001 10
00000000001000001100001010001010100000010100100000000000011000000000001 10
00001001000000000000000000010000000000000000000000000000000000000000 7
010000000000000001000000000101000000100011000000000000000000000000000 1
00000000000000000001000010100000000000000010010000010000010000000000001 10
0000100100000000000000000001000000100000000001000001000000001000000000110 7
100000000000000000000000000000000100000001010000100000000000000000000 3
000000000000000000000000000000000111000000000000000010000001100000001000010 4
010000000000000001000000000101001000000011000010000000000000000000000 1
000010010000000000000001001000000000001000001000001000000001000000000100 7
00000000000000000000010000101010010000000000010010000000000001000000000011 10
00100000100100000011000100000000100000000000000000000000000000000000 2
00000010010000000000110000000000000000000000000000000000000000000000 5
00000000000000000000010000101000000000010000010010000000000001000000000001 10
001000011101000000010001000000001000001000001000001000000000000000000 2
1000000000000000000000000000000001010000010100001010010010000000000110000000 3
00100000100100000001000100000000100000000000000000000000000000000000 2
00000000000000011000000000001000010000000000000000100000100000100010000000 8
0000000000001110000000000000000100000000000000100000110000100010000000 8
00000100000000000000000000000000011100000000000000000000000000000000 4
00000010010000000000110000000010000000000000000000001110000000010000001000 5
00000000000000000000000000000000011000000000000000000000000000000000 4
0000000000000000010100001010000000100000000100100000000000110000000000001 10
0000000000000000000000000000000001000010110000000000000001001000000000001 10
00000000000000000000000000000000011000000000000000000000000000000000 4
00000000000000011100000000000000000000000000000000000000000000000000 8
000000000000000000000000000000000100000100100000001000100000000000000 10
00100000100100000001000100000001010000000000000000000000000000000000 2
100000000000000000000000000000000100000000101000010000000000000000000 3
00000001000000000000000000000000011000000000000000000000000000000000 4
01000000000010000100000000001011010000000110000000000000000000000000 1
10000000000000000000000000000000010000000010100001000000001000000000 3
10000000000000000000000000000000010000000000000000000000000000000000 3
000001100000100100110000000000000001011000000010000000100000000000000 9
00100000100101000000110010000000001000000000000000000010011000000000010000000 2
00001001000010000000000000010000000000000000000000000000000000000000 7
00001001000000000000000000000000000000000000000000000000000000000000 7
00000000000000000000000000000000011000000000000000000000000000000000 4
00100000000000000000000000000000010000110000000000000000000000000000 4
10000000000000011000000000000000000000000000000000000000000000000000 8
10001001000000000000000000000000000000000000000000000000000000000000 7
00010000000101000000000000000000000000000000000000000000000000000000 6

0001000000100100100000000000000010000100000000000101100000000010000 6
000001000000100100100100000000010110100000000100000000100000000000000 9
0000000000000000000000000000000011000000001000000010000001100000001000010 4
010000000000100010000000001110000000000110000000000000000000000000000000000 1
010000000001000001000001000101000000010011000000000000000000000000000000000 1
0010000010010000000100010000000001000000000000000000010100001000000000000 2
001001000000100100100000000000000010100000000100000000000000000000000001010 9
0000000000100011000000000000000010000000000000000100000100000100010000000 8
10000000000000000000000000000000100010001010000110010000000000000011000000001 3
001000001001000000010001101000001100000000000000000100000000000000000000000 2
010000000000000000000000000000001110001001000110000000000000000000000000000 1
010000000000000000000000000000001010010000000110000000000000000000000000000 1
0010000010010000000100010000000001000000000000000000010000000000000000000 2
100000000000000000000000000000001000000010100001000000000000000000000000000 3
101000000000000000000000000000001100010001010000100000000000000000000000000 3
000010010000000000000000000000001000010000000000000100100100000001000000000100 7
0100000000000000000000000000000011010000001010000000000011000000000000000000000000 1
101000001001000000010001000000010100000000000000000001000000000000000000000 2
00000000000000011000000000000000001001000000010001000001000001000100000000 8
00001001000000000000000000000001010000000000000000001100001100000001000000000100 7
000000000000000110000000000000000010000000000000001000001001001000100000000 8
000000000000000110000000000000000010000000000000001000001000001000100000001 8
00000000000000011000000000000000001000000100000001000001100011000100000000 8
00100000000000000000000000000001000011100000000000000001001000000010000000000001 10
001000001001000000010001010000001000000000000000000010000000000000000000000 2
0000000010000000001000010100000000000000000100100000000000100000000000000001 10
100000000000000000000000000000001000000010100001010000000000000000000000000 3
00000000000000000000000000000000110000000000000000000100000001100000001000010 4
000100000010000000000000000000010000010000100000001000010100000000000000000 6
000000000000000000000000000000011001000000001000011000001100000001000010 4
0000000000000000000000000000000100000000000000000000010000001100000101000010 4
000010010000000000000000000000010000000000000000000001000001000000000000000100 7
0000000000000001100000000000000000100000000000000010000101000100010100010010000 8
010100000010000000000000000000000000000000000000000000000000000000000000000 6
000010010000000000000000000000010000000000000000000100010100000001000000000100 7
00000000000000011000000000000000001000000000000000100000100000100110000000 8
000001000000100100100000010000000010100000000100000000000000000000000000000 9
000001000001100100100000000000000010100000000100000000000000000000000000000 9
000001000000101001000000000000000010100000000101000000000000000000000000000 9
100000000000000000000000000000001000000010100001000000000000000000000000000 3
000001000000100100100000000000000010100000000101010000000000000000000000000 9
000100000010000000000011000000000010000100000000000010100000000000000000000 6
001000001001000000010001000000001000000000000000000000000000000000000000000 2
000001000000100100100000100000000001010000100010000000000000000000000000000 9
001000001001000000010001000000001000000000000000000000000000000000000000000 2
001000001001000000010001000000001000000000100000000100000000000000000000000 2
0000010000001001011000000000000000100101010000001000000010011000000000000 9
00001001000000000000000000000000100000000000000010010000010000000011100 7
00101001000000000000000000000000100000000000000001000001000000001000000000100 7
00001001000000000000000000000000100000000000000001000001000000001000000000100 7
00001001000000000000000000000000100000000000000001000001000000001000000000100 7
000100000010000000000000000000000000000000000000000000000000000000000000000 6
000000100100000000001100000000000000000000000000000000000000000000000000000 5
000001000000100100100000000000000001010010000010000001000001000000000000 9

```

0000100100000000000000001000000000000000100010100001001000000000100 7
0000000000000000000010000101000000000000000100100000000001000000000001 10
0001000000100000000000000000000000001000010000000000101010000000000110000 6
100001000000100100100000000000000010111000000001000000000000000000000001 9
0000000000000110000000000000000010000000000000100001100000100010000001 8
010000010000000010000000001010000000001100000100000000000101000100000 1
0000000000000100000000000000001100000000000000010000001100000001000010 4
0000010000001001001000000000000000101000000001000000000000000000000000 9
0000100100000000100000000100000000100000010100000100010001000000000100 7
0000000000000110000000000010100100000100000000100000110000100010001000 8
0000001001000000000011000000000000000000000000000000000000000000000000 5
000000000000000010000000000001100000000000000000000000000000010000010 4
0000011101010000000011000000100000000000000000000000000000000000000000 5
00000000000000000000000000000000000010001100000000001000001000000110000001000010 4
0000000000000011000000000000000000000000000000000000000000000000000000 8
00000000000000000000000010001111000000000000000010010000000000101000000000001 10
000010010000000000000000100100000010100001001100000100000001100000000100 7
00100000100100000001000100000000110000000000000000000000000000000000000 2
01000000000000000100000000001010000000000110000000000000000000000000000 1
0001000000000011000000000000000000000000000000000000000000000000000000 8
0000001001000000000011000000000000000000000000000000000000000000000000 5
01000000000000000100000100010100000000001100000000100000000000000000000 1
0010000010010000000100010000000001000010000010000000000000000000000000 2
00100000100100000001000100000000010000000001000000001000000000000000000 2
0000000000000011000000000000000000000000000000000000000000000000000000 8
0001000000100100000100000000000000000000000000000000000000000000000000 6
0000001001000000000011001000000000000000000000000000000000000000000000 5
0000100000000011000000001000000001000000000000000000000000000000000000 8
01000000000000000100000100101001000000011000000001000000000000000000000 1
0001000001001000000100010000000001000010000010000000000000000000000000 2
00100000100100000001000100000000010000000001000000001000000000000000000 2
0000000000000011000000000000000000000000000000000000000000000000000000 8
0001000000100100000100000000000000000000000000000000000000000000000000 6
0000001001000000000011001000000000000000000000000000000000000000000000 5
0000100000000011000000001000000001000000000000000000000000000000000000 8
01000000000000000100000100101001000000011000000001000000000000000000000 1
0000100100000000000000000000000000000000000000000000000000000000000000 7
0000000000000000000000000000000000000000000000000000000000000000000000 4
01000000000000000100000000010100000000011000000000010000000000000000000 1
0001000000100101000000000000000000000000000000000000000000000000000000 6
1000000000000001100000000000000000000000000000000000000000000000000000 8
01000000000000000100001000010100000000011000000000000000000000000000000 1
0001000000100000000010000000000101000010000001000001010000000000000000 6
0100000000000100010000000001010000000001100000000000000000000000000000 1
1000000000000001100000000000000000000000000000000000000000000000000000 8
0100000000000001100000000000000000000000000000000000000000000000000000 8
0000010000001001001000100000000000101100000001000000000000000000000000 9
0000000000000011000000000000000011000000000000000000000000000000000000 8
0000000000000011001000000000000000000000000000000000000000000000000000 8
1000000000000000000000000000000000000000000000000000000000000000000000 3
1000000000000000000000000000000000000000000000000000000000000000000000 3

```

30

34

29

32

24

26

31

37

28

29

This represents the output from the ART algorithm in Appendix I when presented with the data in Appendix III. The two numbers at the end of each symptom vector line are the numerical cluster name assigned by the algorithm and the original cluster name. Note that the two names do not coincide because the data are random with respect to the underlying clusters. That is, the ART algorithm recognizes clusters in the order in which it detects them, and then assigns a name beginning with the maximum number of clusters (20 in this run) and decrements the name by 1 for each new cluster detected. For example, original syndrome number 4 is the first cluster detected, so it is assigned number 20 by the ART algorithm. Next, original syndrome number 6 is detected, and it is assigned number 19, and so on. At the end of the data set is a tabulation of the number of patients detected in each of 10 clusters (followed by ten 0s, indicating that the algorithm did not establish any of the up to 10 additional clusters it could have), followed by the sum of these numbers (300, indicating that every patient was found in a cluster, that is, there were no "unclassified" patients). Finally, the last set of records at the end of the data set are the syndrome prototype vectors along with the corresponding number of patients clustered with each. A comparison of the data in this appendix with that in Appendix III shows that the ART algorithm identified all 10 syndromes correctly, and it correctly clustered all the patients correctly without error.

33

[illegible]

```

0000010000001001001000000000000010101000000001000000001000000000000000 17 9
00000000000000010010000101000000010000000010010000000000100000000000001 15 10
010000001000000010000000001010110000010110000000000000000000000000100000 16 1
0000011000001001001000000000000000101000000010000000000000000000000000 17 9
0000011101000000000011000000000000000000000000000000000000000000000000 13 5
0001000000100000000000000000000000000000000000000000000000000000000000 19 6
0000000000000011010000000000000000000000000000000000000000000000000000 18 8
0000000000000000000000000000000000000000000000000000000000000000000000 20 4
0001100100000000000000000000000000000000000000000000000000000000000000 14 7
0000000000000011000000000000000000000000000000000000000000000000000000 18 8
0100000000000000100000000001010000000000110000000000000000000000000000 16 1
0000010000001001001000000000000000001011000000010000000000000000000000 17 9
00000000000000000000100001010000000000100000010010101000000010000000011001 15 10
1000000000000000000000000000000000000000000000000000000000000000000000 12 3
0100101001000000000001100100000000000000000000000000000000000000000000 13 5
0100000000000000000000000000000000000000000000000000000000000000000000 16 1
0000000000000011000001000000000001000000000000000000000000000000000000 18 8
1000000000000000000000000000000000000000000000000000000000000000000000 12 3
0010000000000000000000000000000000000000000000000000000000000000000000 20 4
0010000010010100000100010000000001000000000000000000000000000000000000 11 2
0100000000000000000000000000000000000000000000000000000000000000000000 16 1
0100000000000000000000000000000000000000000000000000000000000000000000 16 1
0000000000000000000000000000000000000000000000000000000000000000000000 15 10
0010000010011000000100010000000001010000000000000000000000000000000000 11 2
0000010000001001001100000001000000101001000001000000000000000000000000 17 9
0000000001000000000000000000000000000000000000000000000000000000000000 20 4
0000001001000000000001100000100000000000000000000000000000000000000000 13 5
0010000010010000001100010000000001010000000000000000000000000000000000 11 2
0000000000000000000000000000000000000000000000000000000000000000000000 20 4
0000000000000000000000000000000000000000000000000000000000000000000000 20 4
0000100100000001000000000000000000000000000000000000000000000000000000 14 7
0000000000000000000000000000000000000000000000000000000000000000000000 15 10
0000000000000000000000000000000000000000000000000000000000000000000000 18 8
0000000000000000000000000000000000000000000000000000000000000000000000 20 4
0000000000000000000000000000000000000000000000000000000000000000000000 20 4
0001000001100000000000000000000000000000000000000000000000000000000000 19 6
0000000000000000000000000000000000000000000000000000000000000000000000 20 4
0000000000000000000000000000000000000000000000000000000000000000000000 18 8
0100001000000000000000001010100000000001100000100000000000000000000000 16 1
0000000000000000000000000000000000000000000000000000000000000000000000 15 10
0000001001000000000001100000000000000000000000000000000000000000000000 13 5
0000100100000000000000000000000000000000000000000000000000000000000000 14 7
0010010000001001001000000000000000000000000000000000000000000000000000 17 9
0000010000001001001000000000000000000000000000000000000000000000000000 17 9
0000010000001011001000000000000000000000000000000000000000000000000000 17 9
0001000100100000000000000000000000000000000000000000000000000000000000 19 6
0010000010010000000100010001000010000000000000000000000000000000000000 11 2
1000000000000000000000000000000000000000000000000000000000000000000000 12 3
0010001010010010000100010000000000000000000000000000000000000000000000 11 2
0000000000000000000000000000000000000000000000000000000000000000000000 15 10
0010000010010000000100010000000000000000000000000000000000000000000000 11 2
0100000000000000000000000000000000000000000000000000000000000000000000 16 1
1000000000000000000000000000000000000000000000000000000000000000000000 12 3
0000100100000000000000000000000000000000000000000000000000000000000000 14 7

```

[illegible]

[illegible]

[illegible]

[illegible][illegible]

Appendix V. Prototype Graphical User Interface

The visual basic program interface allows the user to enter symptoms by an input file, or directly on the screen for symptom vectors having up to 15 components. It displays the proportion of input assigned to each class as the height of bars. The number 15 was selected to be large enough for experimentation and demonstration purposes and small enough to allow an uncrowded screen.

ART1 Classifier

Classifies Vectors according to ART1 Algorithm

data	class
0	0
0	0
1	1
0	0
0	0
0	0
1	0
0	0
0	0
0	0
1	1
1	1
0	0
0	0
0	0

Vigilance factor: .65
Number of Categories: 5
Number of symptoms: 15

Buttons: w ready, Initialize, Show cat's, show output, use data, use file

Number in each class

number of data in each class

MATCH: .75
WINNING INDEX: 3

EXIT

4/8/95 RW

successful classification

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE JULY 95		3. REPORT TYPE AND DATE COVERED FINAL
4. TITLE AND SUBTITLE Dynamic Aggregation of Symptom Data			5. FUNDING NUMBERS Program Element: 63706N Work Unit Number: M0095.005-6412	
6. AUTHOR(S) Angus, J.E.; R Williamson, D Pick. et al.				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Health Research Center P. O. Box 85122 San Diego, CA 92186-5122			8. PERFORMING ORGANIZATION Report No. 95-34	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Naval Medical Research and Development Command National Naval Medical Center Building 1, Tower 2 Bethesda, MD 20889-5044			10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) The health and well-being of U.S. service personnel assigned to duty in foreign countries is of vital importance to the successful completion of the intended missions. Upon relocation, such groups represent nonindigenous populations exposed to the endemic diseases of the region and other hazards associated with the assignment. Field epidemiologists and preventive medical specialists need ways to obtain and assess clinical data as quickly and accurately as possible and to relate the various signs and symptoms at the time of presentation into associated categories to support early detection of disease and illness. The objective of this study was to develop a computerized algorithm that would accept, sequentially in real time, patient symptom vectors and dynamically identify patterns of probable syndromes/diseases. The approach taken in this study was that provided by Adaptive Resonance Theory (ART). A simple ART algorithm was developed that was capable of sequentially accepting patient symptom vectors and dynamically clustering them into patterns or syndromes described by prototype symptom vectors. The basic ART algorithm was demonstrated with simulated symptom data from 300 patients representing 10 underlying syndromes. The algorithm was shown to correctly identify the syndromes and to cluster the patients into their correct syndromes.				
14. SUBJECT TERMS MEDICAL SURVEILLANCE ADAPTIVE RESONANCE THEORY (ART)			15. NUMBER OF PAGES 40	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT Unlimited	