The Automatic Identification of Aerospace Acoustic Sources

by

Randolph H. Cabell

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Mechanical Engineering

APPROVED:

Dr. Christopher R. Fuller, Chairman

Dr. Walter F. O'Brien

Dr. Alfred L. Wicks

February, 1989

Blacksburg, Virginia

The Automatic Identification of Aerospace Acoustic Sources

by

Randolph H. Cabell

Dr. Christopher R. Fuller, Chairman

Mechanical Engineering

(ABSTRACT)

This work describes the design of an intelligent recognition system used to distinguish noise signatures of five different acoustic sources. The system uses pattern recognition techniques to identify the information obtained from a single microphone. A training phase is used in which the system learns to distinguish the sources and automatically selects features for optimal performance. Results were obtained by training the system to distinguish jet planes, propeller planes, a helicopter, train, and wind turbine from one another, then presenting similar sources to the system and recording the number of errors. These results indicate the system can successfully identify the trained sources based on acoustic information. Classification errors highlight the impact of the training sources on the system's ability to recognize different sources.

Acknowledgements

I would like to thank Dr. Christopher Fuller and Dr. Walter O'Brien for giving me the opportunity to work on a research project as interesting as this one. I would also like to thank Dr. Alfred L. Wicks for serving on my committee. Most if this work was completed at NASA Langley Research Center in the Applied Acoustics Branch. I am thankful to all the members of that branch for their interest in the project and technical advice, as well as the NASA Langley Research Center for its financial support of this work.

I must thank and Blue Thunder, for making life so interesting at Buckroe Beach.

A big round of thanks goes to those friends and family who opened their homes to me when I needed a place to stay, especially . I would have been cold and penniless without your hospitality. My family deserves thanks for their encouragement and help, but also for their patience and understanding. All of you helped in more ways than you can imagine. Special thanks goes to Mom and Dad for starting me on the right track from the very beginning. I wish you both luck in your new lives.

Finally, my most sincere thanks to the one person who has a chance of understanding everything, . Let's go to the beach when this is all over.

Table of Contents

List of Illustrations

1. Introduction

The blending of traditional fields of science with contemporary areas of research creates novel approaches to old problems. In particular, the union of acoustics studies with artificial intelligence, such as an intelligent system that is trained to recognize acoustic signatures, has the potential for many applications. An acoustic signature is the characteristic noise emitted by some source in normal operation. For example, the signatures of transportation vehicles are heard everyday, from the noise of a passenger car to the roar of a jet airplane. All machinery emits some kind of acoustic radiation and a great deal of research is undertaken each year in order to describe, control, or regulate such radiation.

The application of an intelligent recognition system to the identification of acoustic signatures has many implications. Non-destructive evaluation of the operating characteristics of rotating machinery, for example, could be facilitated by an intelligent recognition system that is taught to recognize the onset of part failure. More conventional identification techniques, such as radar, could be supplemented with acoustic information. Such information would offer independent evidence as to the identity of an unknown source.

This work describes an acoustic identification system that could be used for the above applications, but has been tailored to distinguish the signatures of aerospace acoustic sources. In this application,

an intelligent identification system trained to recognize relevant sources could facilitate monitoring and enforcement of noise regulations. The system could identify aircraft that violate pre-existing noise thresholds, or could be used to monitor traffic to establish noise thresholds. The accurate identification of a noise source would reduce confusion from irrelevant sources such as ground vehicles. Incorporating intelligence, or adaptability, increases the generalization capabilities of the system and makes it applicable to dynamic problems. Adaptability means the system does not have to be manually adjusted to account for deviations in the problem; instead the normal operation takes into account occasional variations and changes in the sources under consideration.

The use of pattern recognition techniques in an intelligent monitoring system has been investigated in few previous articles. A noise pollution recognition system is described in [1], but is limited in the number of sources it can identify. A system similarly limited in scope was designed to identify helicopters based on their radiated noise [2].

Other examples of acoustic recognition systems can be found in the field of non-destructive evaluation. These systems are taught to recognize part failure through analysis of the acoustic environment. This technique has been applied to analyzing structural noise for discriminating crack growth emissions from other extraneous emissions [3]. Another NDE system is used to monitor acoustic emissions from reactor feeder pipes [4].

Intelligent recognition systems are also used for speech recognition. Speech recognition systems typically learn to recognize a string of phonemes, restricted either in the number of phonemes or in the number of speakers they can understand. Although speech systems are acoustic in nature, most work involves the calculation of probabilities of occurrence of strings of phonemes using Markov chains.

Pattern recognition techniques are used on a much broader basis than just speech recognition and acoustic identification. These techniques are used in optical character scanners such as bank check readers and supermarket registers to identify scanned images. Other applications include medical

diagnosis systems that classify a patient's symptoms into a disease category, sometimes suggesting tests to validate the diagnosis.

1.1 Pattern Recognition

In general, pattern recognition systems use evidence from several sensors to determine the membership of an unknown pattern. The evidence is tailored to the problem, so that a medical diagnosis system might use body temperature and white blood cell count, for example, whereas a supermarket scanner might use the time-voltage output of an optic sensor.

An intelligent recognition system learns the best way to weigh the evidence in order to classify the patterns. For example, the supermarket scanner could be taught to recognize characters. One way to do this would be for the system to scan a certain character, recording the output of the optic sensor. The recorded output would be identified as belonging to that character. This is done for each character in the machine's alphabet, so that when finished, the machine would simply compare the output of an unknown character with those it has stored in memory.

1.2 Identification of Acoustic Sources

This thesis describes the design and testing of an intelligent recognition system used to identify acoustic sources based on evidence accumulated from the output of a single microphone. In particular, the system was trained to distinguish the noise signatures generated by several jet planes, propeller planes, a helicopter, train, and wind turbine. The signature of each source was the noise

produced by typical operation of each source; a plane flyover, a train moving on the tracks, or the wind turning the wind turbine.

This recognition system was designed to test the ability of pattern recognition techniques to distinguish aerospace acoustic sources. For this reason, the scope of the problem was limited. Several different types of jets and propeller planes were used for training, but only one helicopter, train, and wind turbine were used. The results of this study will indicate whether or not a system using pattern recognition techniques can distinguish several different airplanes from the noise produced by a single helicopter, train, and wind turbine. A more extensive study would be required to generalize the results to all planes, helicopters, etc.

The system was trained to recognize the signatures using sound recordings of members of each class of source. Unlike the training of the supermarket scanner, the system was presented several types of certain sources, i.e. ten different jet planes were used, from a T-38 military trainer, to a 747 jumbojet, yet they all fell under the class of jet planes. This meant the system could not just store the signature of each source because there was significant variation within each type of source.

The results of training were then evaluated by having the trained recognition system identify similar recordings of the five sources. The operation of the system was evaluated using the percent of sources identified correctly.

The recordings of the sources used to test the operation of the system were not exactly the same as those used for training, and in some cases they were substantially different. The number of available recordings of each type of source determined how much variation there was in the test data. In the case of the helicopter, wind turbine and train, the small number of source recordings meant the testing recordings were similar to the training recordings. However, in the case of the propeller and jet planes, the large number of available recordings meant there were significant differences between the training and testing source recordings.

1.3 Thesis Layout

The exact details of the training and testing are discussed later in this thesis, as well as a discussion on pattern recognition theory in general. Chapter 2 gives a brief synopsis of some pattern recognition theories, including Bayesian decision theory and cluster separation theory. Included is a discussion of the training of a system and how a smart system is easier to work with and has wider application.

Chapter 3 describes the components of the system that was used to identify the five different sources. This chapter covers the decision theory, training algorithm, and feature selection algorithm, which was a smart component intended to ease the system's application. In addition, Chapter 3 describes two different architectures of the system to be compared in the testing phase.

Details of applying the system for identifying acoustic sources are discussed in Chapter 4. The training data and testing data are decribed here, including the similarities and differences between the two. Also described are the features, or evidence, in this case, used to describe the signatures of the sources. The signal processing used in analyzing the source recordings is also included in Chapter 4.

The results of testing are covered in Chapter 5. The results are given in the form of percentages of test sources identified correctly by the system, broken down by type of source and system architecture.

Chapter 6 contains conclusions based on the effectiveness of the various parts of the system in identifying aerospace acoustic sources. Recommendations on how to improve the system are given in Chapter 7.

This system is similar in purpose to those used for recognition of underwater sources. Unfortunately, the theory and applications of such systems are of a restricted nature, so they are not included in this discussion.

2.0 Pattern Recognition Theory and Algorithms

The operation of a pattern recognition system can be broken down into a series of steps, shown in Fig. 1. There are two modes of operation; training and classifying. Training is used to teach the system to recognize patterns from different classes using known patterns, while classifying uses the results of training to identify unknown patterns. Training must be done before any classifying because the system first must learn how to recognize patterns from different classes. Once the system has adapted to the pattern classes, the decision logic is stored to be used in classifying unknown patterns.

Both steps require data acquisition on the patterns to be identified. The data acquired on the patterns is then processed by a preprocessor, whose output goes to a feature calculator. In data acquisition, physical measurements are made on the patterns using appropriate sensors, and the measured quantities are then transformed by the preprocessor. For both steps the output of the preprocessor is then used to calculate quantities containing information useful for identification. These quantities are called features, and can be thought of as the evidence used to classify patterns, as described in the introduction.

During the training step, the classification system learns to identify the different classes using a teacher. For most systems, the teacher is a human user that has access to additional information

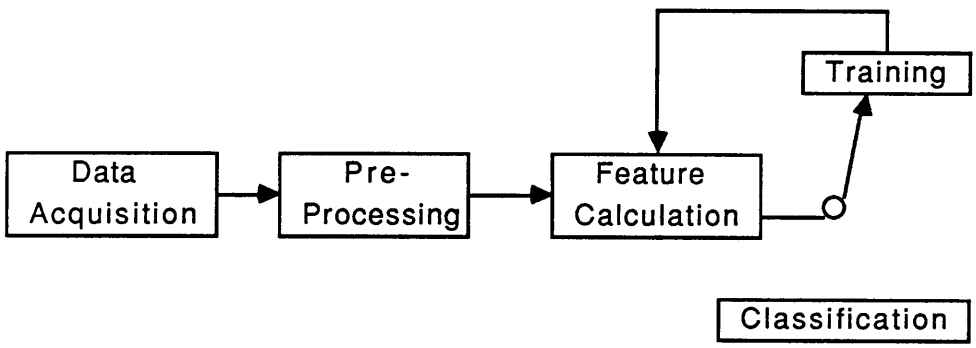

Figure 1. A Pattern Recognition System

about the identity of the patterns presented to the classifier. For example, the human user could have previously recorded aerospace sources and identified them visually. This teacher presents a recording to the classifier along with the class membership of that recording. The classifier adjusts itself to identify that recording correctly. Many different recordings are used so that the classifier will be able to identify recordings with variation, instead of just one type of recording, like the supermarket scanner in the introduction.

In addition to giving the class membership of the patterns, the teacher can also suggest features that might be useful for identifying the classes. As it is difficult to tell which features will be most useful for a given classification task, especially when the features are correlated to some degree, an intelligent system will automatically pick and choose among the suggested features until it has a set that best discriminates the patterns.

Once the teacher is satisfied with the performance of the classifier, training is ended and the system saves the best features and the best way to use those features for classifying patterns. The system is now ready to use this information to classify unknown patterns.

The operation of the system described in the previous paragraphs requires decision logic that uses features to identify patterns, a training algorithm to adjust that decision logic, and a feature selector to choose the best set of features. This chapter discusses possibilities for each of these parts.

2.1 Decision Logic

The pattern recognition problem is not exactly recognition in the familiar sense, rather "the problem of recognition is really just one of classification - classifying unknown vectors onto one of a finite number of categories [5]." Tou and Gonzalez word their definition somewhat differently, saying

"the problem of pattern recognition may be regarded as one of discriminating the input data, not between individual patterns but between populations, via the search for features or invariant attributes among members of a population [6]."

The vectors referred to in the preceding paragraph contain the evidence useful for classification, and the job of the classifier is to use that evidence to assign the pattern to a known category. There are many different methods of using the evidence or features to classify patterns, a few of which are discussed below. See references [6,7] for a more complete discussion.

Throughout this work several terms are used in describing the pattern recognition problem. *Class* refers to a subset of the population whose members share certain properties, *pattern* is used to denote individual members of a class, and *feature* is used to describe those quantities calculated from measurement information to describe the patterns under consideration.

1. Membership Roster - This method is also referred to as template matching [6]. An input pattern is compared to members of a set of stored patterns one by one and the closest match determines its class. Figure 2 shows an example of template matching where the input pattern is a shape which is compared with stored shapes using optical sensors. Whichever silhouette gives the best match, in this example based on light transmitted through an opaque template, determines the class of the input shape. This is similar to the method used by the supermarket scanner described in the introduction. The optic sensor's output is recorded and subsequent characters are identified by comparison with a library of outputs. This method requires near perfect patterns that can be compared with reference patterns, and is sensitive to noise.

2. Common Property Concept - This methodology assumes that patterns belonging to the same class possess similar invariant attributes, making a checklist type of recognition possible. As an example, a biologist traveling in the Amazon jungle discovers an animal and would like to classify it as mammalian or reptilian. The members of each of these categories possess certain invariant characteristics (with very few exceptions), so the biologist need only check off the

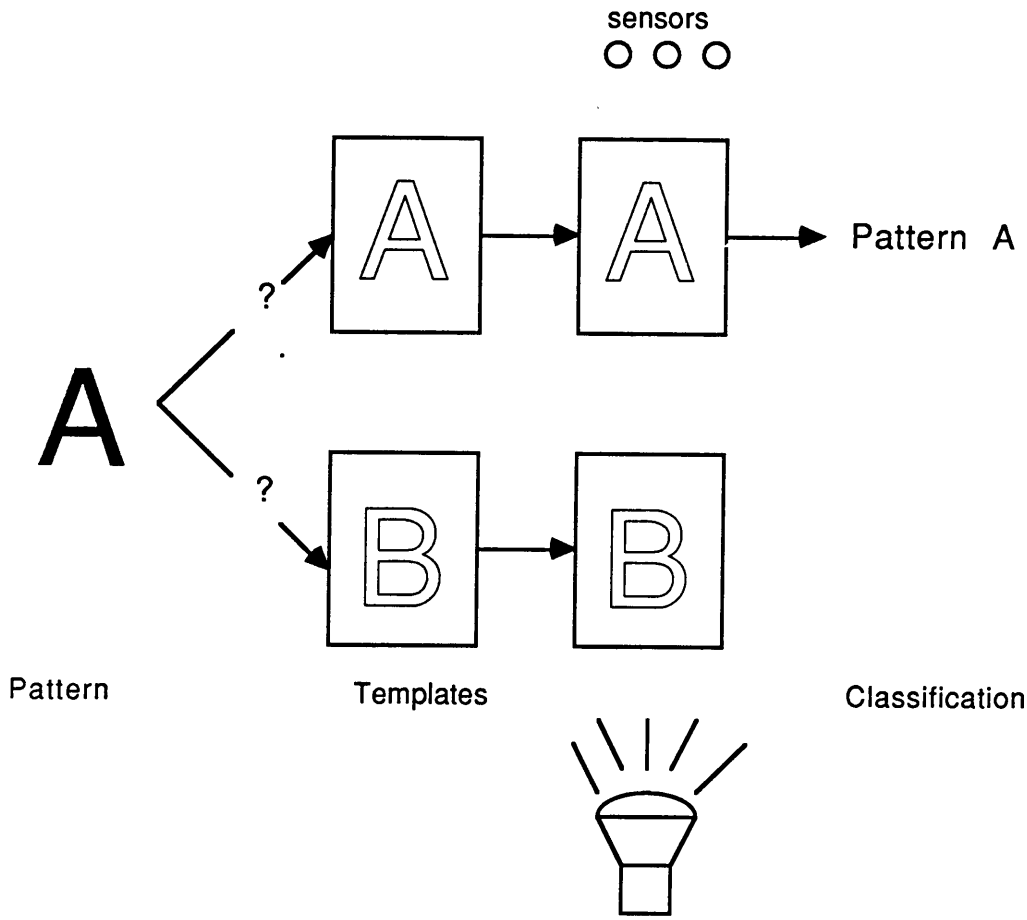

Figure 2. Pattern Recognition by Template Matching

characteristics of her animal. Is it egg-laying or does it have live born offspring? Does it have a 6 or 1-boned lower jaw? An animal meeting the first characteristic in each group would be classified as reptilian, while one meeting the second would be classified as mammalian. For some problems the presence of these invariant attributes is obvious. However, for many applications a set of invariant attributes would be difficult to find, assuming such a set even exists.

3. Syntactic Recognition - Syntactic pattern recognition is a language-oriented approach to recognition using grammars to describe patterns [6,7,8]. This theory breaks patterns up into subpatterns or primitives, and uses grammars to describe the way the subpatterns fit together to make up the whole. Figure 3 shows how syntactic recognition could be used to describe the structure of a chromosome [7]. The chromosome is shown at the top of the tree and it is broken up in a hierarchical manner until it is resolved into a combination of four basic shapes, called primitives. The chromosome can then be described by the string of characters representing these primitives; babcbabdbabcbabd, in this case. Recognition becomes a matter of putting this string in a category. Patterns must have a clear structure for this method to work, so most applications involve pictorial images such as Chinese characters, chromosomes, or particle collision photographs [6].

4. Clustering Concepts - This method characterizes classes by their clustering properties in Euclidean space. Features, calculated to describe the patterns, are used to make up a pattern vector containing the features. These vectors locate the patterns in feature space, and patterns belonging to the same class should form clusters, if the features are properly specified. Decisions on class membership are made based on the relative geometrical arrangement of class clusters [6]. As an example, consider a system designed to classify a set of athletes as being either basketball players or jockeys [6]. Measurements, such as height and weight, would be made on the patterns (athletes) and the height and weight of each athlete could be plotted in a height-weight space. The vectors corresponding to the basketball players would form one cluster, while those for the jockeys would form a different cluster, as shown in Fig. 4 [6]. Once

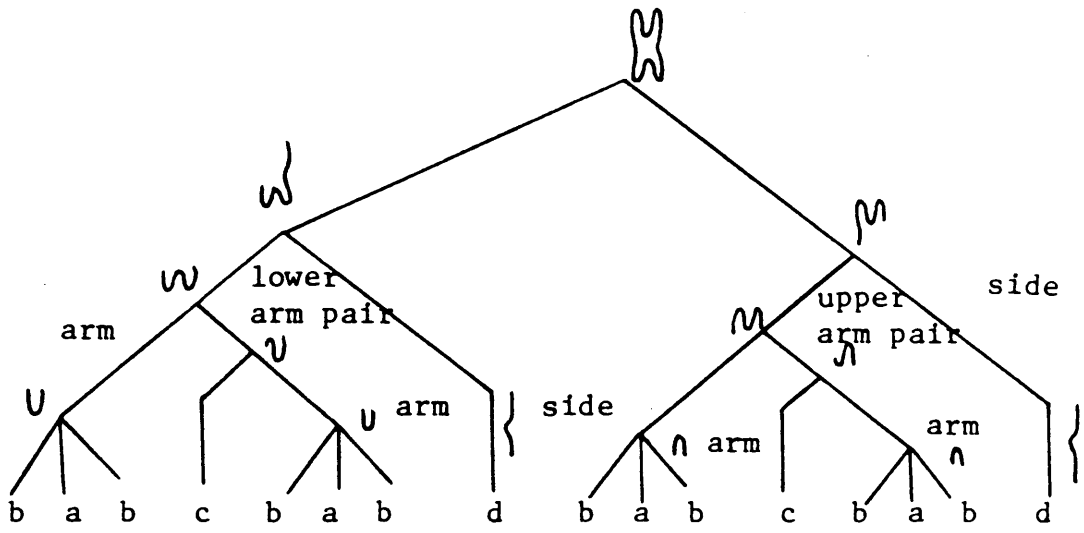

Figure 3. Structural Analysis of a Chromosome [7]

the class clusters are defined, some geometrical decision logic could be used, such as placing a line between class clusters. An unknown pattern lying above the line would be considered a basketball player, while one below the line a jockey.

Of the methods described above, clustering methods were chosen as the most suitable for the acoustic recognition problem. Clustering methods do not require perfect or near perfect patterns, allowing for the variation inherent in acoustic signals. A single acoustic signal can be described by an *n*-dimensional pattern vector, generating clusters in *n*-dimensional space.

Once class clusters have been generated, the problem becomes one of separating those clusters from one another. Methods of separating the class clusters can be grouped into three general categories; distance functions, likelihood functions, and general decision functions. These methods are described in the next three sections.

2.1.1 Distance Functions

Distance functions are the most intuitive of the separation methods, as a pattern's class membership is determined by proximity. An unknown pattern vector is classified as belonging to the class of its nearest neighbor, or nearest neighbors. This method is simple to apply as long as one can define a suitable distance measure, but the classes to be separated must have good clustering properties [6]. Classes with poor clustering properties may yield unsatisfactory results if the shape of their clusters is irregular. A class cluster in the familiar bell shape of a gaussian distribution would be well suited for distance measures, because a pattern lying close to a given cluster is most likely a member of that cluster's class. On the other hand, if a class cluster is cigar shaped, a pattern lying close in one direction may be a member while one lying close in another direction may not be a member.

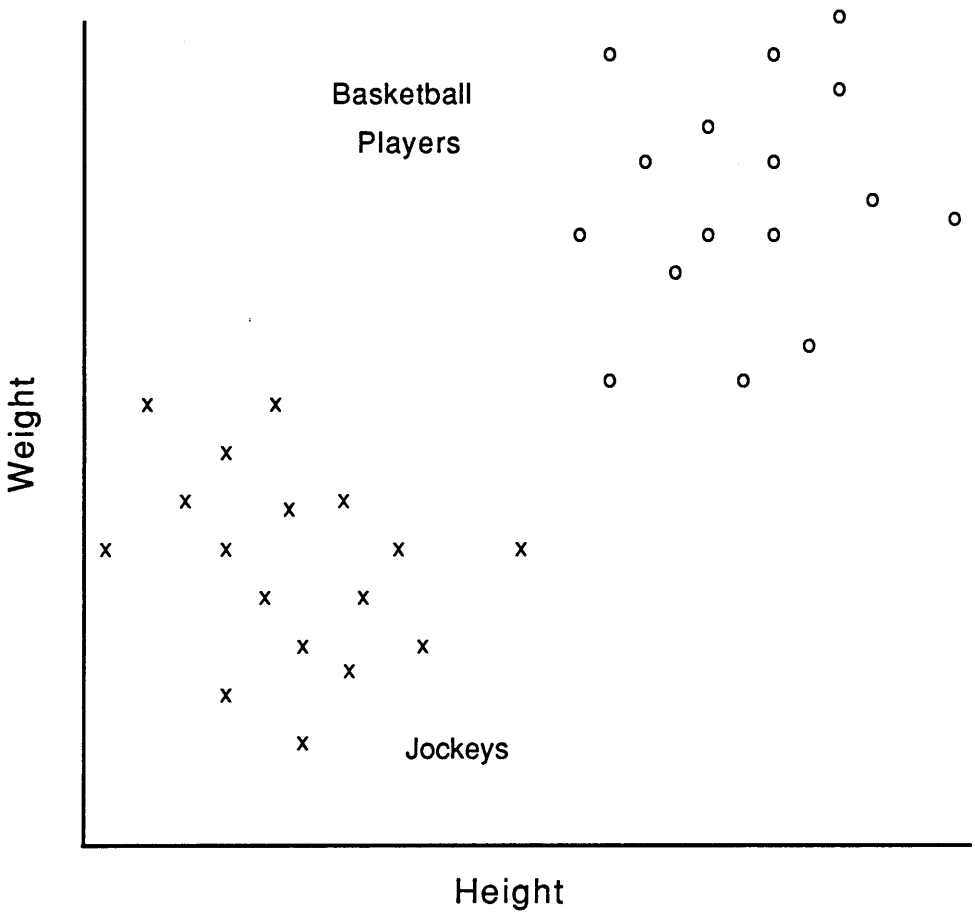

Figure 4. Clusters of Basketball Players and Jockeys [6]

The nearest neighbor method can be expressed mathematically as $\bar{s}_i \in \{\bar{s}_1, \bar{s}_2, \ldots, \bar{s}_n\}$ is the nearest neighbor of $\bar{x}$ if

$$D(\bar{s}_l, \bar{x}) = \min\{D(\bar{s}_l, \bar{x})\} \quad l = 1,2,\ldots,n \qquad (2-1)$$

where D is any distance measure over the pattern space [6], $\bar{s}_i$ represents stored pattern vectors, and $\bar{x}$ represents the unknown pattern vector. This method is particularly useful when classes can be represented by prototypes, such as mean values. In this approach only one representative for each class has to be stored and only one distance calculation for each class is necessary.

If a class cluster is oddly shaped, like the cigar example above, then a single prototype is not representative of the class, yielding incorrect classification for some patterns. In this case, all members of the class would have to be stored, since no one prototype can describe the behavior of that class. The distances then have to be computed to every stored member to find the nearest neighbor of the unknown pattern, which becomes cumbersome for real problems.

This method can be extended to multiple neighbors, where the class membership of the majority of an unknown's neighbors determines its membership. For example, suppose a pattern lay somewhere between two clusters. Distances could be computed to its 5 closest neighbors, and the membership of the majority determines the class of the unknown. This technique would be useful for the cigar shaped clusters discussed above, but again the computations can become complex as the number of stored patterns and nearest neighbors increases.

2.1.2 Likelihood Functions

The statistical approach to pattern recognition was used in the earliest systems and is still used today [9,10]. These methods use the statistical properties of the classes under consideration to determine the membership of an unknown pattern. Such a classification rule will yield the lowest

probability of classification error on average, making it a useful standard for comparison. The statistical nature of likelihood functions means they can be useful even when class clusters are overlapping. By taking into account a class prototyope and the class distribution, a pattern can be assigned to the class to which it is most likely to belong. In contrast, a distance function method uses only the proximity to the prototype, ignoring the distribution of the class.

Likelihood functions require knowledge of the class distributions, or the ability to derive a functional representation of the distribution. This requires a large sample of patterns that accurately represent the class populations. Examples of recognition systems based on statistical techniques are found in the medical domain, where a large data base allows accurate calculation of statistics. Statistical techniques are applied in verifying a hypothesis, for example, in detecting evoked response signals amid background brain activity [9], or to make a complex diagnostic decision based on the state of a patient [10].

In making decisions, statistical classification systems aim to minimize the average risk or loss in making a decision [6]. The average risk in making a decision is defined below, where the symbol ω_i is used to denote class i, and $\bar{x}$ is used to denote a pattern vector. If $\bar{x}$ is presented to the classifier, where $\bar{x} \in \omega_i$, and the classifier assigns $\bar{x}$ to class ω_j, then a loss L_{ij} is incurred. Pattern $\bar{x}$ can belong to any one of M classes, so the expected loss in assigning $\bar{x}$ to any ω_j is

$$r_j(\bar{x}) = \sum_{i=1}^{M} L_{ij}\, p(\omega_i \mid \bar{x}) \qquad (2-2)$$

where $p(\omega_i \mid \bar{x})$ is the probability that $\bar{x}$ belongs to ω_i.

Using Bayes' formula [6],

$$p(\omega_i \mid \bar{x}) = \frac{p(\omega_i)p(\bar{x} \mid \omega_i)}{p(\bar{x})} \qquad (2-3)$$

Eq. 2-2 becomes

$$r_j(\bar{x}) = \frac{1}{p(\bar{x})} \sum_{i=1}^{M} L_{ij}\, p(\bar{x} \mid \omega_i)\, p(\omega_i). \qquad (2-4)$$

The $p(\bar{x})$ term can be dropped from Eq. 2-4 since it is constant with respect to the index i for the same pattern $\bar{x}$.

Normally the loss L_{ij} is assumed 1 for incorrect classification and 0 for correct classification, or $L_{ij} = 1 - \delta_{ij}$. Incorporating this assumption into Eq. 2-4 results in

$$r_j(x) = \sum_{i=1}^{M} (1 - \delta_{ij})\, p(\bar{x} \mid \omega_i)\, p(\omega_i) \qquad (2-5)$$

$$= p(\bar{x}) - p(\bar{x} \mid \omega_j)\, p(\omega_j). \qquad (2-6)$$

This result is used for classification by evaluating the losses for each j, and assigning $\bar{x}$ to ω_i if

$$p(\bar{x}) - p(\bar{x} \mid \omega_i)p(\omega_i) < p(\bar{x}) - p(\bar{x} \mid \omega_j)p(\omega_j) \qquad (2-7)$$

or

$$p(\bar{x} \mid \omega_i)\, p(\omega_i) > p(\bar{x} \mid \omega_j)\, p(\omega_j) \quad j = 1,2, \ldots M;\ j \neq i. \qquad (2-8)$$

If we define

$$d_i(\bar{x}) = p(\bar{x} \mid \omega_i)p(\omega_i) \qquad (2-9)$$

then $\bar{x}$ is assigned to ω_i if

$$d_i(\bar{x}) > d_j(\bar{x}) \quad j = 1,2,\ldots M;\ j \neq i \qquad (2-10)$$

which is a decision function we can use to determine the class of a given pattern [6].

A statistical classification rule is powerful because it takes into account the behavior of the pattern populations in determining an optimal decision boundary. However, it is often quite difficult to make an assumption concerning the population distributions. Unless the patterns are noisy variations of a well defined prototype, the use of a classical distribution such as normal, binomial, or Poisson is not advisable [11]. The behavior of real patterns cannot always be described by simple distributions, so a pattern recognition system using them will not yield meaningful results.

Likelihood functions are most useful for differentiating similar vehicles under similar operating conditions. This property could be used, for example, to identify the type of helicopter, once another classifier has determined that an unknown source is a helicopter.

2.1.3 Decision Functions

The term decision functions refers to the separation of class clusters using appropriate mathematical functions or surfaces used as decision boundaries in n-dimensional Euclidean space. Decision functions can be generated using statistical information as shown in Eq. 2-10, but in this section the decision functions are derived without statistical information. This type of pattern separation has been used in a noise pollution recognition system [1], a reactor pipe emissions monitoring system [4], and for distinguishing crack growth emissions from flight noises in the structure of a plane [3]. Their widespread application is due to their power, versatility, and ease of training.

The basic concept behind a decision function is that class clusters can be separated by placing a boundary between the classes. Figure 5 shows a line decision function used to separate the basketball and jockey clusters discussed earlier. Pattern vectors on the positive side of the decision function or decision surface belong to the population of basketball players, while those on the negative side are jockeys. An error in classification occurs if the pattern vector of a jockey lies on

the positive side of the decision surface or that of a basketball player lies on the negative side. The decision function must be placed to minimize such errors.

The boundary used to separate the clusters can be quite complicated, including nonlinear combinations of feature values, for example the height squared, in the basketball player-jockey example. The chances of separating two sets of patterns increase as the nonlinearity of the decision function increases [6]. One can visualize a case where two clusters are not separable by a linear decision surface but are separable by a quadratic or cubic surface. The types of decision surfaces reported in the literature depend on the clusters to be separated, including ellisoids [12] and extremely nonlinear surfaces calculated to optimize classification [3].

The simplest form of a decision function is a linear decision function applied to the 2-class case, shown in Fig. 5. Its equation in n-dimensional pattern space is

$$d(\bar{x}) = w_1x_1 + w_2x_2... + w_nx_n + w_{n+1} = \bar{w}'\bar{x}, \qquad (2-13)$$

where $\bar{x} = \{x_1, x_2, x_3,..., x_n\}$ is the pattern vector. The feature values are given by $x_1, x_2 ....$ The weight vector $\bar{w} = \{w_1, w_2, ..., w_n, w_{n+1}\}$ contains the coefficients of the decision function.

The evaluation of this decision function results in a scalar value. One possible implementation for classification is

$$\bar{x} \in \omega_1 \ \textit{if}\ d(\bar{x}) > T \qquad (2-14a)$$

or

$$\bar{x} \in \omega_2 \ \textit{if}\ d(\bar{x}) < T \qquad (2-14b)$$

where T is a threshold value. Equation 2-14a reads pattern $\bar{x}$ is assigned to class ω_1 if the value of the decision function, $d(\bar{x})$, is greater than a preset threshold T.

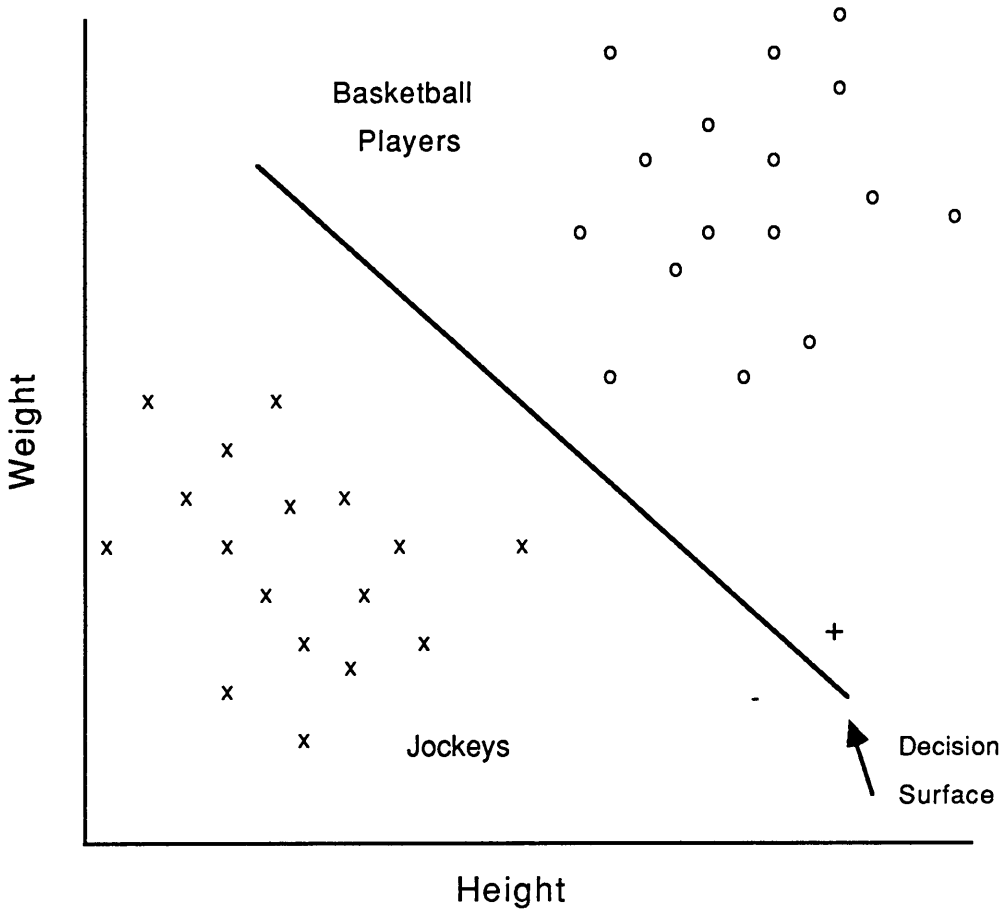

Figure 5. Decision Function Separating Basketball Players and Jockeys

A decision function linear in the feature values such as Eq. 2-13 encompasses all types of nonlinearities by virtue of a simple transformation. Consider a decision function given by

$$d(\bar{x}) = w_1 f_1(\bar{x}) + w_2 f_2(\bar{x}) + \ldots + w_n f_n(\bar{x}) + w_{n+1} = \sum_{i=1}^{n+1} w_i f_i(\bar{x}). \qquad (2-15)$$

Let

$$\bar{y} = (f_1(\bar{x}), f_2(\bar{x}), \ldots, f_n(\bar{x}), 1) \qquad (2-16)$$

which makes the decision function linear in $\bar{y}$, so that

$$d(\bar{y}) = w_1 y_1 + w_2 y_2 + \ldots + w_n y_n + w_{n+1} = \sum_{i=1}^{n+1} w_i y_i. \qquad (2-17)$$

Any decision function, no matter how nonlinear in the feature values, can be treated as linear using this transformation [6].

Once the form of the decision function has been set, it must be placed with respect to the classes to be separated. For the two class case, one must consider what value of a threshold in Eqs. 2-14a and 2-14b will give optimum performance, while for multiple classes one must consider how many functions to use. Decision functions can be placed with respect to class clusters in one of three ways [6].

- Case 1. Each pattern class is separable from all other classes by a single decision surface. This means that for M classes there are M two-class decision functions; either the pattern belongs to the class or it doesn't.

- Case 2. The classes are pairwise separable so there is a decision function separating every pair of classes. This requires $M \times (M-1)/2$ decision functions (M classes taken 2 at a time). A pattern is classified in class ω_i if $d_{ij}(\bar{x}) > 0$ for all $j \neq i$.

- Case 3. For M classes, there are M decision functions such that $d_i(\bar{x}) > d_j(\bar{x})$ for $x \in \omega_i$ for all j, $j \neq i$. This is a variation of case 2, since each decision function in case 2 can be derived by subtracting pairs of decision functions of case 3.

Once the structure of the decision functions and the number to be used has been determined, they must then be placed to separate the class clusters. This requires the calculation of the weight vector of Eq. 2-13 such that the surface separates the clusters. The coefficients of the weight vector can be determined in 2-dimensions by visualizing the placement of a separation surface, but it is impossible to use visualization in higher dimensions. Training algorithms solve this problem by determining coefficients of the decision surfaces separating the classes using a set of training patterns.

2.2 Training Algorithms

Decision function separation of pattern classes is straightforward in theory, but in practice their use requires determination of a large number of coefficients describing multi-dimensional separation boundaries. Multiple dimensions rules out using simple visualization to place the decision functions, so training is used to find the coefficients of the decision surface or surfaces using representative patterns. Training provides pattern recognition algorithms with greater accuracy, wider applicability and simplify their application to complex problems [13].

The type of learning described here is called supervised learning since a 'teacher' tells the system which patterns belong to which classes. Unsupervised learning uses the classifier to generate clusters given a set of unknown patterns [6].

The history of learning algorithms dates to the 1940's and 50's from work done on models of human learning. Researchers in that period were trying to formalize general models of learning for networks that could be implemented on an analog or digital simulator [14].

Around 1960 F. Rosenblatt proposed a device called a perceptron as a simplified model of the human neural system [14]. Rosenblatt's perceptron was a binary pattern associator that was meant to be a crude model of the structure and learning capabilities of the human nervous system [14].

In the basic perceptron, an array of sensory inputs is connected to an array of associative units, which are in turn connected to a response unit. This structure is shown in Fig. 6 [15]. The sensory units can be optical, for example, activated when their field of view is occupied by an object. Groups of sensory units are connected to associative units, the A array in Fig. 6. The outputs of these associative units are weighted and summed in the response unit resulting in a scalar output.

The output of the response unit is a linear sum of the weights multiplied by the output of the associative units,

$$R = \sum_{i=1}^{n} w_i x_i = \overline{w}'\overline{x}. \qquad (2-18)$$

This result is nothing more than an implementation of a linear decision function (see Eq. 2-13).

Along with the perceptron Rosenblatt proposed a training algorithm to adjust the weights on the connections until the response unit's output was correct for a given input. In terms of the linear decision function, this corresponds to finding the weight vector that gives correct classification for all patterns, assuming the patterns are separable by a linear decision surface. The learning scheme,

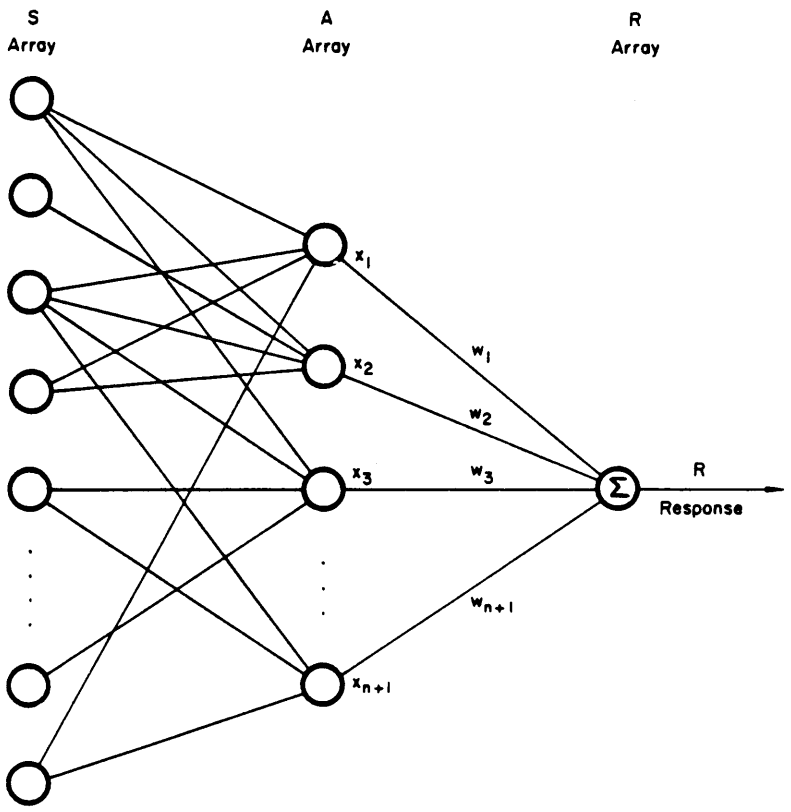

Figure 6. Single Layer Perceptron [15]

known as the perceptron algorithm, was based on a reward-punishment concept that Rosenblatt proved was guaranteed to converge to a solution, if such a solution exists [6,14]. The training procedure given in Eqs. 2-19 - 2-21 is for two classes, where the members of class 1 lie on the positive side of the decision surface and members of class 2 on the negative side of the surface. The algorithm is an iterative algorithm, where at the k^{th} training step a pattern $\bar{x}(k)$ is presented to the classifier. If $\bar{x}(k) \in \omega_1$ and $\bar{w}'(k)\,\bar{x}(k) \leq 0$, then $\bar{x}$ has been classified incorrectly, so the weight vector is adjusted according to

$$\bar{w}(k+1) = \bar{w}(k) + c\bar{x}(k). \qquad (2-19)$$

If $\bar{x}(k) \in \omega_2$ and $\bar{w}'(k)\,\bar{x}(k) \geq 0$, then $\bar{x}$ again has been classified incorrectly, so a new weight vector is calculated using

$$\bar{w}(k+1) = \bar{w}(k) - c\bar{x}(k). \qquad (2-20)$$

If $\bar{x}$ is classified correctly, the weight vector is not changed for that iteration,

$$\bar{w}(k+1) = \bar{w}(k). \qquad (2-21)$$

The magnitude of the correction to the decision function is determined by the constant c in Eqs. 2-19 and 2-20, typically given the value 1. The weight vectors can be initialized to any value at the start of training, although it is customary to start with all coefficients at 0.

The coefficients of the weight vector, $\bar{w}$, are adjusted by the amount $c\bar{x}(k)$, which is either added to or subtracted from the current value of the weight vector. In Eq. 2-19, the pattern $\bar{x}(k)$ was incorrectly classified in class ω_2 because the value of the decision function was less than zero. To correct this mistake the weight vector is changed by adding a multiple of the pattern vector $\bar{x}(k)$.

The same situation occured in Eq. 2-20, except the value of the decision function was too high, so the amount $c\bar{x}(k)$ is subtracted from the weight vector.

The value of c in the gradient formula, Eq. 2-22, is a correction increment, which is a constant in the perceptron algorithm. Any value c in the perceptron algorithm will lead to convergence as long as the value 1 does so, since any other value merely scales the patterns without changing their separability. Subsequent references to the perceptron algorithm assume c to be 1.

The training of a classifier to recognize two classes using the above algorithm, requires the presentation of several members of each class, which make up a *training set*. At each iteration, a pattern from the training set is presented, the decision function is evaluated, and its result is compared with that of the teacher. If no error was made, training continues with the next pattern in the training set. If an error was made, a correction is made according to Eqs. 2-19 or 2-20, and training continues with the next pattern. Training does not stop until the decision function correctly classifies all members of the training set, or until a sufficiently low error rate has been attained. A normal training run requires several cycles through the entire training set before a solution is reached. This is especially true if there are a large number of patterns in the training set, and if those patterns belong to class clusters that are not widely separated.

An alternative method of training a classifier is to use a *training set* to adjust the values of the weight coefficients and an *evaluation set* to evaluate the calculated decision surface. Training would be the same, except the evaluation set would be used to determine when training is done, i.e. training doesn't stop until all members of the *evaluation set* are classified correctly or classified with a small enough error. The evaluation set is used to simulate realistic conditions. It would be expected that the classifier could correctly identify the training set since all adjustments on the decision surface are based on this set. The evaluation set insures the results of training are applicable to a wider range of patterns, not just those used for training [16]. A similar approach to this latter case will be used in this thesis.

This algorithm is part of a larger class of adjustment algorithms based on gradient concepts [6]. Examples of other gradient algorithms are the Widrow-Hoff algorithm and the Grossberg algorithm [14]. Gradient techniques seek a solution by minimizing a criterion function using the formula

$$\overline{w}(k+1) = \overline{w}(k) - c\left\{ \frac{\partial J(\overline{w}, \overline{x})}{\partial \overline{w}} \right\}_{\overline{w} = \overline{w}(k)} \qquad (2-22)$$

where $J(\overline{w}, \overline{x})$ is a criterion function.

The perceptron algorithm can be derived using this formula and the criterion function

$$J(\overline{w}, \overline{x}) = \frac{1}{2}(|\overline{w}'\overline{x}| - \overline{w}'\overline{x}) \qquad (2-23)$$

The partial derivative of this function is

$$\frac{\partial J}{\partial \overline{w}} = \frac{1}{2}[\overline{x}\, sgn(\overline{w}'\overline{x}) - \overline{x}] \qquad (2-24)$$

where $sgn(\overline{w}'\overline{x})$ is 1 if $(\overline{w}'\overline{x}) > 0$ and $sgn(\overline{w}'\overline{x})$ is -1 if $(\overline{w}'\overline{x}) \leq 0$. Substitution of the partial derivative from Eq. 2-24 into equation 2-22 gives

$$\overline{w}(k+1) = \overline{w}(k) + 0 \quad if\ \overline{w}'(k)\,\overline{x}(k) > 0 \qquad (2-25a)$$

and

$$\overline{w}(k+1) = \overline{w}(k) + c\overline{x}(k) \quad if\ \overline{w}'(k)\,\overline{x}(k) \leq 0 \qquad (2-25b)$$

where $c > 0$. This is the perceptron algorithm given in Eqs. 2-19 - 2-21 [6].

Other gradient techniques are based on different criterion functions, and the number of possible techniques is limited only by the number of possible criterion functions [6].

The perceptron algorithm is simple yet powerful in its implementation. It allows the determination of the coefficients of a multi-dimensional plane, or hyperplane, separating sets of classes by presenting known patterns to the classifier. The perceptron does have shortcomings as a model of the human neural system and as a complex pattern recognizer. During the late 1960's a rigorous

mathematical study of the perceptron showed that it was not powerful enough for a great number of applications unless the number of units was increased beyond practical limitations [14,15].

Despite its shortcomings, the algorithm is useful for finding the coefficients of a simple decision function. Recent research into human learning has concentrated on multilayer perceptrons, known as neural networks, where the output of the associative arrays is a non-linear function of the inputs [14].

Learning algorithms for statistical decision functions require learning the mean vectors and covariance matrices of the classes with an assumed distribution. There are methods for generating a functional representation of the class distributions [6] but these methods are beyond the scope of this thesis.

2.3 Feature Selection

The generation of class clusters requires the representation of each pattern by a feature vector. Features are calculated to describe the patterns, and if properly chosen, vectors of patterns belonging to the same class will cluster near each other and away from other classes. The specification of a good set of features is crucial to the success of the classifier; the real power of a classifier comes from carefully chosen features [5]. Carefully chosen features produce strongly disassociated class clusters, which improve the performance of any classifier [11]. The selection of a good set determines the classifier accuracy and sets bounds on error rates [5,6,11,16,17].

This problem is known as the feature selection problem. System specific knowledge is not always available to use in prespecifying an optimal set of features, and even if it were avaliable, it is possible to overspecify features, introducing too many constraints for the classifier to work on real data.

The purpose of an automatic feature selector is to allow the user to specify a large list of features that might be useful for classification, from which the system builds an optimal set for the problem at hand. This frees the user from the tedious task of specifying a good set of features for every recognition problem the classifier encounters. Instead, a large list containing a broad range of features is specified once, and the selector takes care of choosing the set that works best on one particular problem.

Methods for accomplishing this selection can be grouped into transformational and search techniques. These two approaches are described below. See references [6,16] for a more complete discussion.

2.3.1 Transformational Selection Techniques

Transformational feature selection techniques map the features into a lower dimensional space [6,16]. One method of reducing the features is to increase the orthogonality of the classes, which usually involves the calculation of eigenvalues. An example of one such technique is the Karhunen-Loeve expansion, found in the references [6,16].

These techniques become quite complex for large numbers of features [6,16], and work best when the features are independent of each other. The acoustic source recognition problem involves both a large number of features and features that are correlated with one another, since they are all derived from the output of a single microphone.

2.3.2 Search Selection Techniques

Search techniques use a measure of the discriminatory power of a set of features to choose the best set from a candidate list. Different search methods are available depending on how much time is to be spent finding a good set of features. Most search techniques do not guarantee an optimal set of features, just a good set of features. Usually the time and computational commitment required for an optimal search are not justified compared to the results of a suboptimal technique [9].

A short list of search techniques includes *exhaustive*, *accelerated*, and *sub – optimal* searches [16]. An *exhaustive search* requires evaluating all possible combinations of features, which becomes impractical for even a small number of features. An *accelerated search* takes advantage of certain properties of the evaluation criterion, but again can be quite time consuming [16].

Sub – optimal methods are the most widely used, encompassing a wide range of approaches to the problem. The most common sub-optimal techniques are listed below.

Sequential Forward Selection: SFS is a building up technique that starts with no features in the best set. One feature at a time is added to the set and if it improves recognition performance then it is kept in the best set. This technique is sub-optimal because once a feature has been added to the final set it is never removed [1,16].

Sequential Backward Selection: SBS is similar to SFS except that all features are present in the best set at the start and are removed one at a time. If the removed feature does not impair recognition performance then it is left off the list. This technique is sub-optimal because once a feature is removed from the list it is never put back on the list [16].

There are variations of the SFS and SBS technniques, most notably the *plus* q *– take away* r method. This method adds q features, tests the classifier, then takes away r features that do not

affect performance significantly. This routine can be forward or backward building depending on the relative values of q and r.

The *n best* method picks the n best features from an ordered list to make up the best set. The features are ordered according to a specified criterion, such as a statistical measure of separation. The problem with this technique is that the features are evaluated one by one and no measure is made of their performance as a set.

Regardless of the power of the technique, there is still no substitute for 'good engineering' [17]. The best selection technique will come up with a poor set if it is given a poorly chosen candidate set, so it is important for the user to incorporate system knowledge into specification of the candidate set, even though an automatic selector is used.

2.4 Classifier Structure

Much like the different methods of placing decision functions with respect to class clusters, the decision logic can also be arranged in different ways depending on the relationships between the classes to be separated. If there are several classes under consideration and a few of them share certain characteristics, it might be useful to perform a preliminary separation of the few classes from the other classes. This is referred to as hierarchical classification.

Hierarchical classification is, as the name suggests, classification performed on different levels. The classification on each level may use different decision logic or different information, but the purpose is the same; to break down the classification process into a series of intermediate abstractions.

The advantage of intermediate abstractions is especially apparent as the number of classes increases beyond two. An explanation of the advantages of intermediate abstractions is given in section 3.4, where the structure of the classification system is discussed.

Intermediate abstractions useful for acoustic source identification depend on the sources to be identified. If the system is to identify trucks, jet planes, and helicopters, for example, a useful intermediate step would be to separate ground sources from aerospace sources. This step would simplify the aerospace source identification problem, since the aerospace source classifier could optimize class separation without interference from the trucks.

Different levels of classification have been suggested by the authors of a noise pollution recognition system [1]. They suggest using different types of classifiers depending on the characteristics of the sources to be identified. For example, a linear discriminant classifier could be used to identify a source as being a helicopter, then a statistical classifier could be used to identify the type of helicopter. Reference [11] describes a system that makes a decision which is treated as a conjecture. A second part of the system verifies or rejects the conjecture.

Neural networks, previously mentioned in the section on training algorithms, have multiple layers which seem to have the ability to represent intermediate abstractions [5]. Figure 7 [18] shows a simple neural network, where an input layer feeds a 'hidden' layer, which in turn feeds an output layer. The network's exterior structure is very similar to that of the perceptron with the exception of the extra layer in the middle.

The multiple layers only seem to have the ability to represent intermediate abstractions because there has been no rigorous mathematical analysis of neural networks proving this ability, if such an analysis were at all possible. The statement about a network's ability to represent intermediate abstractions is based largely on the network's architecture and limited experiments suggesting such an ability [14,15]. The hidden units in a neural network offer a space for abstractions unlike the perceptron, which has no middle layer for such abstractions. The perceptron maps input values

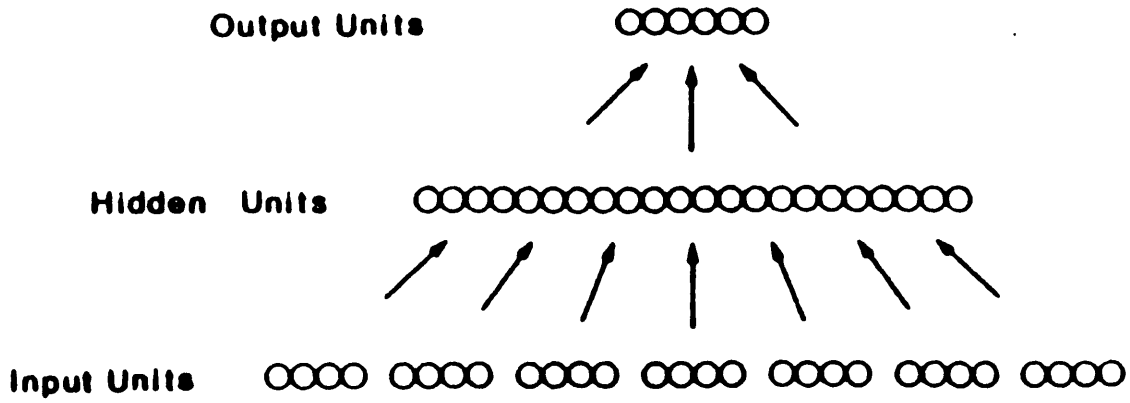

Figure 7. Simple Neural Network Architecture [18]

directly onto output units, making it impossible to generate internal representations. In contrast, the hidden units of a neural network offer a medium in which internal representations can be generated [14].

A discussion of the possibility of internal representations in neural networks and learning algorithms designed to produce such representations is beyond the scope of this thesis, but if there is such a property, it would be useful for classification tasks. The interested reader should see references [14,15] for a more in depth discussion on multilayer perceptrons and neural networks.

3.0 System Design

This chapter describes the components of the acoustic source identification system designed for this study using pattern recognition techniques. The system was designed with the following goals in mind:

1. To recognize a variety of aerospace acoustic sources

2. To recognize these sources under a variety of operating conditions

3. To eliminate the need to pre-specify optimal features

The decision logic, training algorithm and feature selector, discussed in this chapter, reflect the desire to meet these goals.

3.1 Classifier Design

The classifier must take into account the clustering properties of the class populations. If, for example, a statistical classifier is to be used, then it should use a reasonable distribution to describe the class clusters. Accordingly, a first step in specifying a classifier is to describe the clusters that are to be encountered. In this case, the variety of operating conditions of different sources indicated the clusters would not be uniformly distributed around a single prototype. Airplanes taking off sound different from airplanes landing, so a cluster of the class of planes would be poorly represented by a single prototype.

Statistical methods assume the classes can be described by a distribution. However, unless the patterns in the classes are just noisy variations of a well defined prototype, a distribution such as Gaussian, binomial, or Poisson is not a good descriptor and will yield poor results [11]. Real patterns are subject to more than noisy variations about a prototype, especially when a variety of operating conditions is to be encountered [17]. Hence a statistically based classifier would not give optimal results applied to this problem.

Similarly, distance methods require well formed class clusters where proximity can be used to determine class membership. As discussed above, the use of prototypes could impair the performance of the system. To overcome this problem, all training patterns could be stored, and each unknown pattern compared with every training pattern to determine the class membership. However, this method would require a large amount of storage space and cumbersome calculations, making it a poor choice for the acoustic recognition problem.

On the other hand, decision function methods don't require clean clusters or class distributions like the other methods, so it was decided these methods would be used for the classifier. Decision functions do require specification of the form and placement of the functions relative to the classes.

The different placement options were outlined in section 2.1.3. It was decided that case 3 would be used, described below [6].

There are M decision functions, $d_k(\bar{x}) = \bar{w}'_k\bar{x}$, $k = 1,2, \ldots, M$ for M classes. These functions are such that if $\bar{x} \in \omega_i$,

$$d_i(\bar{x}) > d_j(\bar{x}) \quad \textit{for all } j,\ j \neq i. \tag{3-1}$$

The placement of the decision functions in case 3 was most desirable compared with the placement in cases 1 and 2. Case 1 used one decision surface to separate each class from every other class. It is easy to visualize a scenario where this approach would fail, such as a class that is surrounded by the other classes, for which it is impossible to place a single linear decision function between it and every other class.

Case 2 is very similar to case 3 in that a decision surface is used to separate pairs of classes, but case 2 differs in its implementation. Case 2 requires the calculation and storage of $M(M-1)/2$ decision functions (M classes taken 2 at a time)[6]. A pattern is classified as belonging to class ω_i if $d_{ij}(\bar{x}) > 0$ for all $j \neq i$.

In contrast, case 3 requires the storage and calculation of only M decision functions. Classification is determined by the decision function with the highest value. Case 3 is a variation of case 2, since the decision functions of case 2 can be derived by subtracting pairs of decision functions in case 3.

The classification of a pattern $\bar{x}$ using case 3 is shown in Fig. 8 [6]. A pattern in vector form, denoted $\bar{x}$ in the figure, is fed to a preprocessor, which transforms the feature values, resulting in an output $x'_1, x'_2 \ldots$ The transform could be any function computed on the feature values, including quadratic functions of combinations of features. These new features are then fed to a multiplier.

Recall that a decision function $d(\bar{x})$ is equal to the product of a weight vector $\bar{w}$ and a pattern vector, $\bar{x}$. The multiplier in Fig. 8 evaluates each of the M previously trained decision functions, resulting in a scalar value for each one. The maximum selector then chooses the highest valued decision function and assigns the input pattern $\bar{x}$ to that class.

The preprocessor in Fig. 8 illustrates how simple it is to construct a discriminant that is nonlinear in the feature space. Instead of using the original feature values, any conceivable function could be computed from each feature. For example, the value of the first feature, x_1, could be squared, the value of the second feature cubed, the third multiplied by the square of the fourth, etc. This resulting decision function would be highly nonlinear in the feature space.

The multiclass classifier designed with this acoustic recognition system makes no transforms on the features shown in Fig. 8. Instead, the decision functions are evaluated with the original feature values in the $\bar{x}$ vector, resulting in a simple linear discriminant. Although the implementation of a nonlinear decision function is quite simple, the number of computable functions of feature values is endless, and only linear combinations were investigated in this work.

3.2 Training Algorithm

The training algorithm used to determine the coefficients of the M decision functions is an adaptation of the perceptron algorithm described in section 2.2 for the multi-class case. Recall that a decision function $d(\bar{x})$ is equal to the product of a weight vector $\bar{w}$ and a pattern vector, $\bar{x}$, and that there is one decision function for each of the M classes.

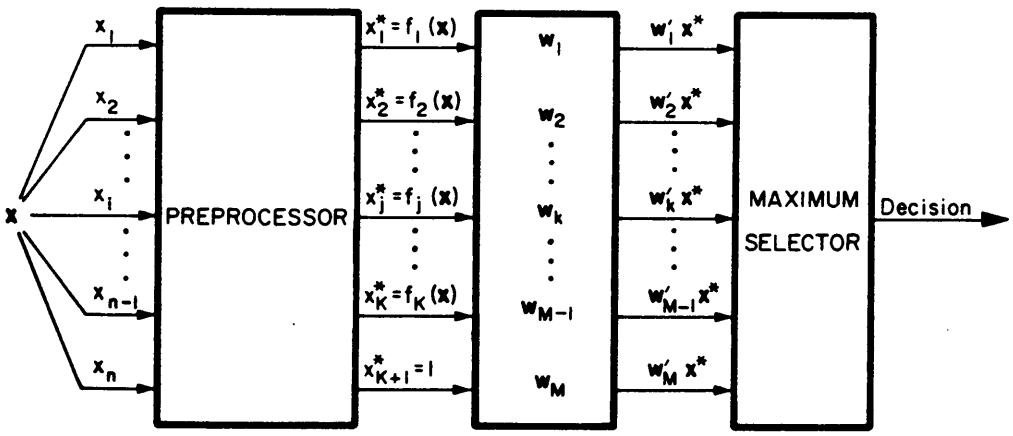

Figure 8. Multiclass Pattern Classifier [6]

This training algorithm is iterative, where at the k^{th} iterative step a pattern $\bar{x}(k)$, belonging to class ω_i, is presented to the classifier. The M decision functions are evaluated by multiplying $\bar{x}(k)$ by each of the M weight vectors. If

$$d_i[\bar{x}(k)] > d_j[\bar{x}(k)] \quad j = 1,2, \ldots, M; \; j \neq i \tag{3-2}$$

then the pattern $\bar{x}(k)$ has been classified correctly and the weight vectors are not changed for the k^{th} iteration,

$$\bar{w}_j(k+1) = \bar{w}_j(k) \quad j = 1,2,\ldots,M. \tag{3-3}$$

If, however, the pattern $\bar{x}(k)$ is classified incorrectly, that is, for some l,

$$d_l[\bar{x}(k)] > d_i[\bar{x}(k)] \tag{3-4}$$

then the weight vectors are adjusted according to:

$$\bar{w}_i(k+1) = \bar{w}_i(k) + \bar{x}(k) \tag{3-5a}$$

$$\bar{w}_l(k+1) = \bar{w}_l(k) - \bar{x}(k) \tag{3-5b}$$

$$\bar{w}_j(k+1) = \bar{w}_j(k) \quad j = 1,2, \ldots, M; \; j \neq i, j \neq l. \tag{3-5c}$$

Note that incorrect classification occurs when one or more decision functions have values greater than the value of the decision function of the training pattern's class. In terms of the above equations, some decision function $d_l[\bar{x}(k)]$ corresponding to class ω_l has a higher value than the decision function of the pattern's real class, $d_i[\bar{x}(k)]$.

The weight vectors that incorrectly classified the pattern are adjusted in Eqs. 3-5a and 3-5b. In Eq. 3-5a, the weight vector corresponding to the correct class of the pattern $\bar{x}(k)$ is adjusted by adding the pattern $\bar{x}(k)$. Equation 3-5b adjusts the weight vectors corresponding to those decision

functions that had values higher than the decision function of the pattern's class by subtracting the pattern $\bar{x}(k)$. Other weight vectors are not changed during this iteration.

This algorithm makes small changes to the weight vectors each time a pattern is incorrectly classified. This requires more than one run through the training patterns to converge to a solution, so the same set of training patterns is cycled through the algorithm until all patterns in the set are classified correctly, or classified with an acceptable error rate.

This algorithm was used with the feature selector as a criterion function, which will be described in the next section, but it should be mentioned that this application usually had high error rates since features were being tested. To avoid infinite iterations of the algorithm due to high error rates, a check was included that prevented the set of training patterns from being cycled through the algorithm more than 200 times. If the 200th cycle was reached and neither a solution nor an acceptable error rate had been reached, then the set of weight vectors that produced the lowest error over those 200 cycles was saved.

3.3 Feature Selection

The features used for classification determine the generalization properties of the classifer and set bounds on the error rates [5,6,11,16,17]. The real power of a classifier comes more from a careful selection of features than selection of the classifier [5], as mentioned in section 2.3. Unfortunately, it is difficult to specify a good set of features for every classification problem, especially when there is a large number of classes requiring a large number of features.

Normally some system specific knowledge is available so that a large candidate list of features with some relevance to classification can be generated. The feature selector included in this system

automatically selects a good set from that candidate list, eliminating the need for extensive study and guesswork by the system user.

The feature selection routine chosen for this system is a search type technique. See section 2.3 for a discussion of feature selection techniques in general.

The feature selection routine was based on the *General Purpose Reduction Intensive* feature selector, or *GPRI*. The algorithm was designed to select a best set of 10 or 20 features from a candidate list in the thousands [16]. It was developed for dynamic signature verification, where a person's signature is classified as valid or forged, but the algorithm is general and adaptable to many problems.

The algorithm uses a variation of the sequential forward selection technique but overcomes the disadvantage of SFS that once a feature is added it is never deleted. The GPRI does this using a 2 step selection. *Step I* builds an initial set of good features, while *Step II* tests new features in combination with the old, adding and deleting as necessary. Features are added or deleted using a criterion to evaluate the performance of the feature sets.

For this study, the criterion used to judge the performance of the feature sets was the error rate of a classifier built with the feature set in question. Each time a feature set had to be evaluated, a classifier was built using a training set of patterns and the perceptron algorithm for the multiclass case, described in section 3.2. This classifier was then evaluated by feeding it patterns of known classification from an evaluation set of patterns and recording how many patterns the classifier identified incorrectly. The fewer patterns misidentified, the better the classifier.

The GPRI algorithm calls for an initial ordering of the features in the candidate set. The author of the GPRI algorithm used an analysis of variance to rank the features from those that described class behavior accurately to those that described it poorly. An initial ordering of the features was not used for this study due to the relatively small number of features and the added complication

of a test. Other changes were made as described below to compensate for the disadvantages caused by leaving out this step.

Step I began with an empty set of best features, pulled the top feature from the candidate list and built a classifier. The error rate of this classifier was stored, then the next feature from the candidate list was added. A new classifier was built with the two features, and if the error rate was less than that of the single feature then the new feature was kept as part of the initial best set. A flow chart of this operation is shown in Fig. 9 [16].

New features were tested in combination with the initial set as shown in Fig. 9, and if they made a significant improvement then they were added to the initial best set. A feature not making a significant improvement was left out of the initial best set and put back at the bottom of the candidate list. The initial ordering ensured the first few features described the pattern classes accurately so the first feature sets generated well separated class clusters. Leaving out the initial ordering as in this study handicapped the first feature sets since they did not necessarily consist of the most descriptive features in the list. Features added to such a set could be punished for not improving recognition when the real problem was the poor feature set to begin with.

Step I as described above was changed slightly from that described in the GPRI reference, to compensate for leaving out an initial ordering. In the original step I, features that did not make a significant improvement had their delete counters incremented by one, then were put back at the bottom of the candidate list. Any feature whose delete counter went above a preset limit was left out of the candidate set and was not tested again. By eliminating the delete counter in step I, features rejected in the early step of selection were not punished, making the selection process longer. This was done to compensate for the lack of initial ordering called for in the GPRI algorithm.

Each time a feature was added, the number of features was checked to see if the initial best set was full, as indicated in the flow diagram. A full best set indicated the desired number of features had

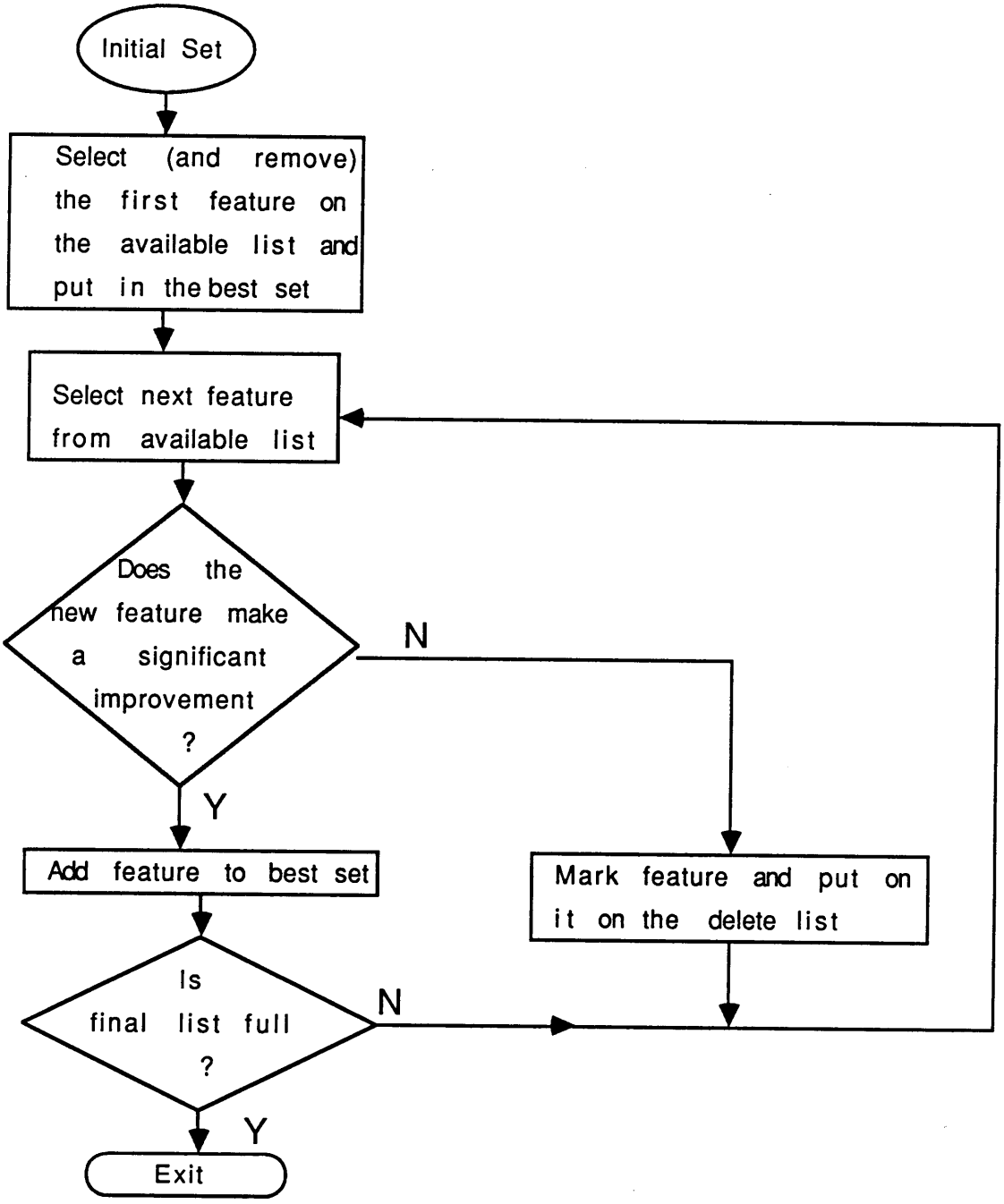

Figure 9. Step I of Feature Selection [16]

been placed in the initial best set. This usually occured before all the candidate features had been tested, so that at the end of step I there were some features in the initial best set, some features tested and replaced at the bottom of the candidate list, and most features still in their original position in the candidate list waiting to be tested.

Step II completed testing on the features left in the candidate set, retesting features that had been replaced at the bottom of the candidate list in step I. Figure 10 [16] is a flowchart of step II. The first part of step II was to select the next available feature from the candidate list. This feature is referred to as the *test feature*. Each feature in the current best set was replaced with the test feature and the new combination evaluated by constructing a classifier with that set and recording its error rate on an evaluation set, as in step I. This is shown in the flowchart by the blocks surrounded by the *set counter* and *counter* = 0 blocks. The counter counted down the features in the best set, while the feature at the current counter position was replaced by the test feature. This continued until the test feature had replaced every feature in the current best set.

When every feature in the best set had been replaced by the test feature, the set with the lowest error rate was selected as the new best set. This new best set could have been the same old best set, meaning the test feature did not improve classification in place of any features, or the new set could include the test feature. Either way, one feature had been left out as not improving the classification rate.

Features left out for not improving the classification rate had their delete counters incremented by one and were placed at the bottom of the remaining candidate list of features.

Step II was also changed from the original step II of the GPRI algorithm. For this system, the delete counters were not incremented until every feature from the candidate set had been tested once, as opposed to the original step II described above where the delete counters were used from the beginning. When every feature had been tested once, then the delete counters were incremented for features that didn't improve the classification rate. This was another change meant to

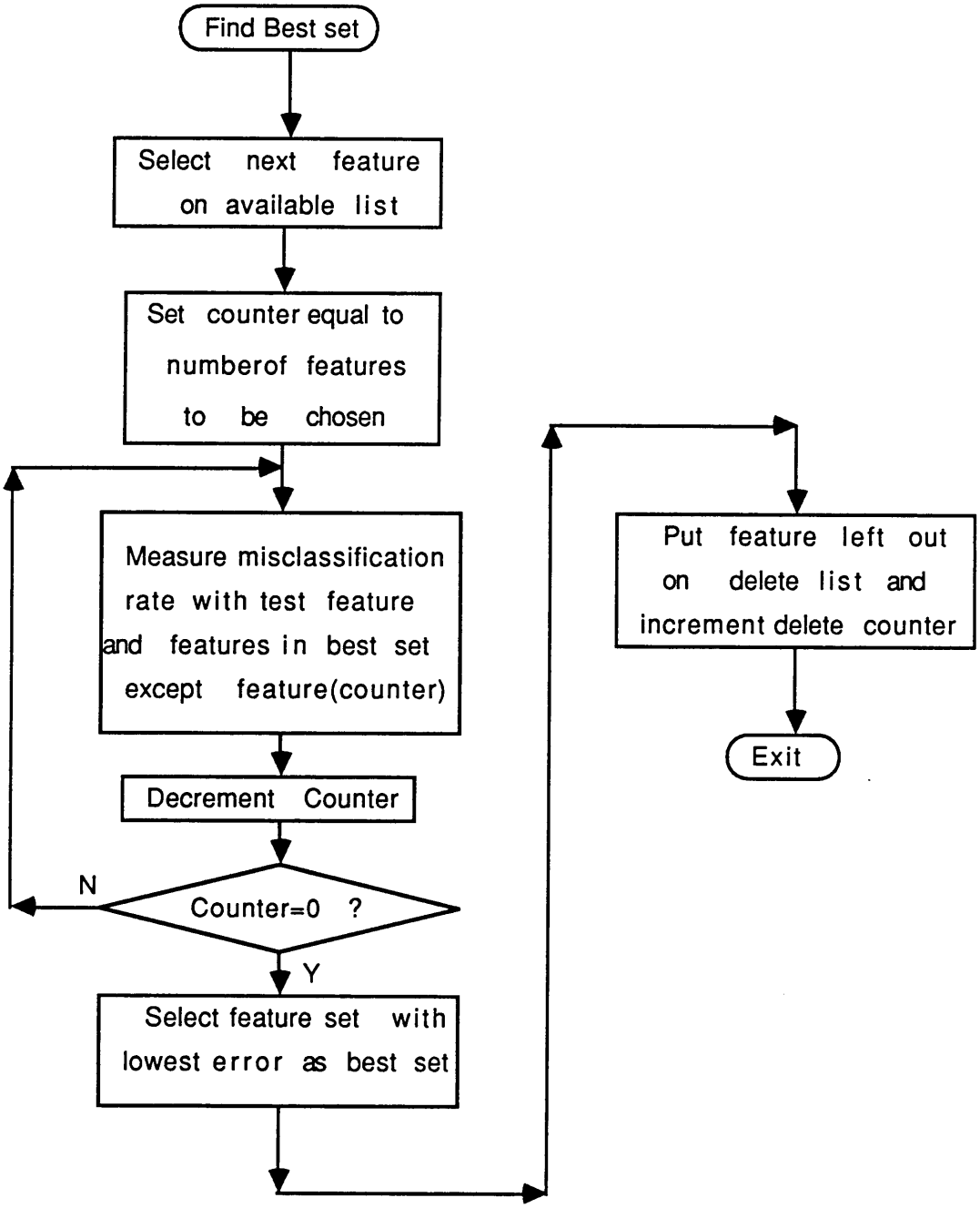

Figure 10. Step II of Feature Selection [16]

compensate for the lack of an intial ordering. Features were not punished for performing poorly until a reasonable feature set had been constructed.

The evaluation of features stopped when all the features left in the candidate list had delete counters greater than a preset limit. The higher this limit the more thorough the search, but also the greater the time for the search. Reference [16] recommended a value of 3, which was used in this system. A limit used to stop feature selection prevented all possible combinations of features from being evaluated, which made the chosen set sub-optimal since its performance was not compared with every possible set. However, the complexity of a truly optimal search is so great that it is impractical to incorporate it in a real system.

Another change was added to the algorithm after a preliminary test to classify acoustic sources was performed. During the tests, the classifier invariably converged to a solution sometime during the search. As more descriptive features were added to the best set, it eventually managed to correctly classify all patterns in the evaluation set. Normally, this would prevent any features from replacing those on the list because no feature can make a significant improvement to a classifier that correctly classifies all patterns. Because of this, the significant improvement qualification was left out, so that any feature added to the set that also correctly classified all patterns was kept.

This could produce an oscillatory type behavior, where one feature in the best set keeps getting replaced by features with related information, especially towards the end of the selection. The new feature sets still must correctly classify all evaluation patterns, so performance is the same on the evaluation set. Whichever feature is in the set when the search is complete is the one that remains in the set. This is a somewhat arbitrary method of concluding the selection, but as long as the feature set correctly classifies all the evaluation patterns then that set must be acceptable.

The major advantage of the GPRI algorithm is that features are evaluated as a set, not individually, using the decision logic of the classifier. In this manner the selected features work well as a set in the framework of the actual classification system.

3.4 Structure

Section 2.4 discussed the benefits of classification logic arranged in a hierarchical manner, one of which was the utility of intermediate abstractions. In terms of acoustic source recognition, this can be thought of as performing preliminary classifications on unknown patterns, such as the separation of ground sources from aerospace sources.

The advantage of using intermediate abstractions can be explained in terms of the properties of the perceptron classifier. This explanation is taken from reference [5]. The perceptron generates a decision surface by multipying the input feature values by a weight vector. A correct decision function is one that successfully separates class clusters, and the perceptron algorithm is a method of using training patterns to converge to a successful decision function.

As the number of classes increases, the algorithm requires more time to converge to a solution. Recall that for M classes there are M decision functions. The algorithm described in Eqs. 3-5a through 3-5c makes changes on a given decision function only when it classifies a pattern incorrectly. For a large number of classes, the corrections to a given decision function will occur periodically, since it will not incorrectly classify every input pattern. This increases the time needed for that decision function to converge, in terms of the number of presentations of training patterns.

An increased convergence time becomes a disadvantage when both the number of training patterns and the time for training are limited, as they would be in any real system. When both the number of training patterns and the number of presentations of those patterns are limited, the probability of convergence to a solution decreases as the number of classes increases. For these reasons, it was decided to study the performance of a hierarchical classifier that used preliminary classifications on the patterns, reducing the number of classes at any one step.

In order to study the performance of a hierarchical classifier, it was decided to compare its performance with that of a single level classifier, one that made no preliminary classifications.

The single level classifier was easy to specify; an unknown input pattern was directly classified as belonging to one of the classes under consideration without any intermediate classifications. This system is loosely hierarchical in the sense that once a pattern is classified as belonging to a class another type of classifer could be used to further identify the signal. This work examines only the first level of classification, that of determining the class of the source, not that of determining the type of jet plane or helicopter.

The hierarchical classifier was more difficult to specify. Ideally the classes would be split up into groupings with similar characteristics and the decision logic arranged to take advantage of these groupings. The sources to be classified in this study were split up as in Fig. 11 to take advantage of different modes of generation of noise. The ground sources were grouped together; those sources that had fixed-wings were grouped together, etc. A similar structure is described in [2] for classifying a larger variety of sources.

First, a signal was classified as belonging to a ground or an aerospace source. The next classification depended on the results of the first classification. If the signal came from a ground source, then it was classified as being that of either a wind turbine or a train. If the signal came from an aerospace source, then it was classified as belonging to either a helicopter or a plane. Helicopters were not further classified, but the planes were separated into propeller and jet planes.

The performance of the tree structure is dependent on the classifications made at each level. For example, it might be useful to instead distinguish propeller planes and helicopters from jets, since the former are characterized by blade passing frequencies and harmonics more prominent than those of jet planes. The specification of a good arrangement of the structure requires expertise and a knowledge of the source characteristics on the part of the designer.

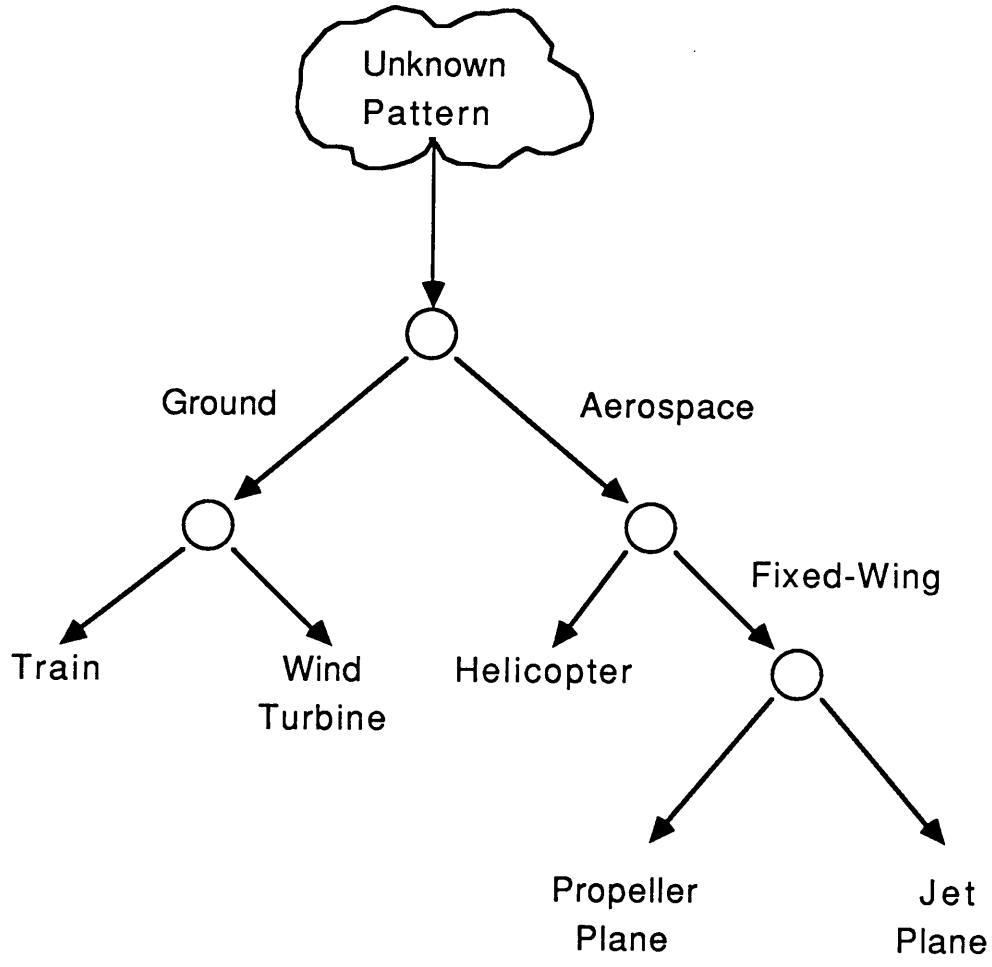

Figure 11. Tree Classification Structure [2]

Both types of structures are only representative of a large number of possible arrangements. These two arrangements were chosen to compare two different approaches. Although both have their merits, one will almost certainly have to implement some type of tree structure to achieve successful results in a system identifying a larger variety of sources.

4. Application to Acoustic Sources

This chapter describes the details pertaining to the application of the acoustic pattern recognition system to aerospace acoustic sources. The system was designed to run on a digital computer, using taped acoustic information obtained from a single microphone located at ground level. The noises generated by the sources were those produced by typical operation of each source; the flyover of an airplane, a helicopter in hover, a train passing by, and the rotation of a wind turbine in the wind.

This study made several assumptions about the noise heard by the microphone. Any motion was on the part of the sources only; the microphone was a stationary observer of the acoustic environment. In addition, the microphone was used to identify noise coming from a single source; multiple sources were not considered. All of the source recordings were made in the environment, so typical outdoor noises were present, such as wind and birds, but none were introduced artificially to confound the identification.

A system used for the identification of acoustic sources could encounter conditions ruled out in the preceding paragraph, but it was decided that they would not be part of this study. Some of the conditions described above could be accounted for using signal processing techniques, such as using a phased-array of microphones to spatially filter incoming noise. The purpose of this study was to

evaluate the effectiveness of pattern recognition techniques in identifying acoustic sources, so the training and testing conditions reflected this philosophy.

The application of the proposed pattern recognition system to aerospace sources involved several details which are covered in this chapter. The application required the digitization of almost two hundred source recordings, processing of the resulting time histories, calculation of features, and the construction of training and evaluation sets. All of the exact details pertaining to the generation of the data base cannot be covered, but this chapter covers the relevant points. These details include signal processing, feature calculation, and training data. In this work these parameters were adjusted to make the system as generally applicable as possible.

Previous work has described setting system parameters to recognize only one type of source [2]. While performance can be optimized for one type of source by setting the parameters correctly, they were set for this system to recognize a larger variety of sources. A system such as that described in [2] could be used at one of the lower levels of classification, as described in Fig. 11, as a specialized classifier, where an incoming pattern has already been identified as a helicopter.

4.1. Acoustic Source Data

The data used for this study were chosen to represent sources encountered in the aerospace acoustic environment. The sources were a helicopter, jet planes, propeller planes, a wind turbine, and a train. The planes and helicopters are examples of aerospace sources that would be encountered in and around an airport. The train and wind turbine sources would not normally be found at an airport, but they are examples of sources that could interfere with a system identifying aerospace sources.

Each of the source classes, helicopter, jet plane, train, etc. had its own unique operating characteristics that could be expressed in terms of acoustic quantities. The chop-chop of the helicopter, the roar of the jets, the hum of the propeller planes, the clicking of the train, and the steady beating of the wind turbine; all are acoustic characteristics that distinguish these sources. It was desired to study the ability of a computer to use such acoustic information to distinguish among the sources.

A data base of noises generated by these sources had to be created for this study. It was decided that magnetic tapes would be the easiest way to store source noises, since tapes were portable and could easily store the information obtained from a microphone. In addition, the output of a tape recorder could be easily connected to a computer where different data records or pattern vectors could be generated by modifying the segments of digital data.

4.1.1 A General Description of the Data Base

The identification system was meant to be general in its application, with the ability to identify a variety of sources. The recognition theory used in the system dictated that the data base used to train the system also be general and include a variety of sources.

Recall from Chapters 2 and 3 that patterns were to be described using n -dimensional vectors containing feature values. These vectors located the patterns in a feature space, and if the features were well chosen then patterns belonging to the same classes would form clusters about each other, and patterns from different classes would tend to be separated. The recognition system put decision functions between these clusters so that an unknown pattern could be identified based on its location relative to the decision functions.

The location of the decision functions was determined using training patterns and a training algorithm. This algorithm placed the decision functions such that they separated the training

patterns without error, or with minimal error. At the end of training, the system had separated the training patterns, stored its results, and was ready to classify unknown patterns.

The decision functions calculated in training are useful only for separating the clusters generated by the training data. The decision functions will be useful for data in general if the training data is representative of the classes in general, but if not, then the system may or may not be able to identify sources different from those in the training set. For example, if multiple recordings of a single type of propeller plane were used during training, then the cluster generated by those patterns would contain operational variations inherent to that particular aircraft. Whether or not that cluster encompassed operational variations of other propeller planes is indeterminate. The system has learned to differentiate that type of propeller plane from the other sources, not necessarily propeller planes in general from the other sources.

The impact of this result on the generation of a training data base is that in order for the system's operation to be general, the training data base must also be general. The incorporation of several different propeller planes in the training data base, for example, will help to insure that the system can recognize propeller planes in general.

With these thoughts in mind, an attempt was made to generate a data base representative of acoustic sources in general. This work was supported by the Applied Acoustics Branch at NASA Langley Research Center, which regularly conducts studies on the generation, propagation, detection, and annoyance of noise produced by aerospace acoustic sources. These studies invariably involved the generation of recordings of such sources during typical operation, so there was a large library of magnetic tape recordings of jets, propeller planes, helicopters, and wind turbines. Not all of the tapes for all of the sources were available, but the content and recording conditions of the tapes used are described below. Each source description also discusses the utility of the available data in generating general representative clusters useful for identifying a variety of sources.

Jet Planes: Recordings of jets were the most readily available of all the sources. Tapes from a NASA annoyance study [19] were used that contained multiple recordings of nine different types of commercial airliners. In addition, several recordings of a military jet were made available for use with the system. The type and number of recordings are given in Table 1.

All of the commercial airliners were recorded on takeoff at an airport, while the military jet was recorded on a straight flyover. Although not all of the recordings were made at the same location under the exact same conditions, they all contained noise characteristic of a jet powered aircraft. The different runs correspond to recordings of different takeoffs or flyovers of varying duration. For example, the set contained recordings of 13 different takeoffs of a 727-100 jet plane. Information was unavailable as to whether or not the 727-100 was the same for each recording, but even if the plane was the same for each recording, it can be assumed that takeoff conditions varied enough between runs to introduce variation in the recorded sound.

The data set of jet planes available for training the system was ideal. The large number and variety of jet recordings meant that the class cluster containing jet planes would be general and account for variation between different types of jets, not just variations in the operation of a single jet. If every class being trained had a data set containing the depth of data and amount of variation of this set, the user could be confident in the classification powers of the system.

Propeller Planes: The propeller plane recordings were also taken from an annoyance study [19] and were recorded at an airport, like the jets described above. The types of planes and conditions of the recordings are given in Table 2. This collection of recordings included both turboshaft and reciprocating engine powered aircraft. The two different engine types were lumped into the propeller plane class as long as the plane was driven by an external propeller.

Most of the propeller planes from this set of tapes were recorded twice, once on takeoff and once landing. Unlike the jets described in the preceding paragraph, multiple recordings of each type of plane were unavailable. This set, being more restricted than the set of jets, wouldn't be able to

Table 1. Description of Jets Used in Data Base

Type	Number of Recordings	Description
727-100	13	Takeoff
727-200	10	Takeoff
737	6	Takeoff
DC-9	7	Takeoff
DC-8 TJ	6	Takeoff
DC-8 TF	7	Takeoff
747	7	Takeoff
L-1011	6	Takeoff
DC-10	13	Takeoff
T-38	8	Flyover

Table 2. Description of Propeller Planes Used in Data Base

Type	Description
King Air	Takeoff
Nord 262	Takeoff
P-3	Takeoff
EMB-110	Takeoff
Dash 7	Takeoff
Viscount	Takeoff
Shorts 330	Takeoff
YS-11	Takeoff
C-130	Takeoff
Swearingen Metro	Takeoff
T-28	Takeoff
King Air	Landing
Nord 262	Landing
P-3	Landing
EMB-110	Landing
Dash 7	Landing
Viscount	Landing
Viscount	Landing
Shorts 330	Landing
YS-11	Landing
C-130	Landing
Swearingen Metro	Landing
Mu-2	Takeoff
Cessna 152	Takeoff
Cessna 172	Takeoff
Supercub	Takeoff
Bonanza IV	Takeoff
G. Tiger	Takeoff

generate a cluster containing as much variation as the set of jets. This cluster would not describe all propeller planes so there is a chance that some propeller planes would be incorrectly identified if not similar to those used to generate the cluster.

The takeoff and landing recordings offered an opportunity to train the system with planes operating under one condition then test the system performance with planes operating under a different condition. This is discussed more fully in the section on training and evaluation data.

Helicopter: There were fewer helicopter recordings available than those of the airplanes. The tapes contained five recordings of a single McDonnell Douglas 500 E helicopter, taken to study the far field propagation characteristics of helicopter noise.

The use of such a limited data base meant that the system was being trained to discriminate only this type of helicopter from the other types of sources. Other helicopter recordings would have to be used in training for the system to recognize helicopters in general.

Wind Turbine: The wind turbine noise was taken from four one minute long recordings of a wind turbine. The recordings were made downwind from the turbine, where the noise is most prominent. The wind turbine was the only one of the sources in this study that did not move relative to the microphone, making the level of each recording constant.

The wind turbine class is unique in that noise is not produced by an engine or vibrations of a structure like the other sources. Instead, a wind turbine is characterized by low frequency blade passing noise. The cluster generated by these recordings would be useful in identifying other wind turbines since their noise signature is relatively uncomplicated.

Train: An empty coal train passing about 10 meters from the microphone was recorded by the authors as part of this study. The train was very long, so noises from the engines were not used, only those sounds produced by the cars on the rails. The sound of the empty cars alone lasts approximately five minutes on the tape.

The noise produced by a train varies from car to car, depending on flat spots on the wheels, squeaky brakes, and other peculiarities of each car. There are however, characteristic noises of a train, so by partitioning the single recording into many smaller units, the cluster should be representative of noises produced by an empty train.

The training and evaluation sets of data had to be drawn from the recordings described above, but some recordings had to be left out so they could be used for the testing phase. The classifier would be expected to classify the members of the training set correctly, but the real test would be whether or not it could identify recordings it had never heard before. For this reason, certain source recordings were left out of the training and evaluation sets for later use in testing the performance of the system.

The precise makeup of these sets is described in the next section.

4.1.2 Training, Evaluation and Testing Data

The recordings described in the previous section made up the complete data base for this study. These source recordings were digitized, preprocessed, and features calculated to describe the sources. Training and evaluation sets were constructed from the processed source recordings, and the test set was constructed from separately processed recordings. This section discusses the details of the generation of each of these sets of data.

The training sets would be used in the perceptron algorithm described in Chapter 3 to find the coefficients of the decision functions. The decision functions separate clusters generated from the training data, so the data must be representative of the class members for good results.

The evaluation sets would be used to evaluate the results of the training sets in the feature selection algorithm, as described in section 3.3. Like the training sets, there must be enough patterns in the

evaluation sets to insure that the decision functions generated in training separate realistic clusters. Some evaluation sets in this study contained the same sources used in training sets when there were not many source recordings available. This would not harm the results of training; only insure that the decision functions were capable of separating realistic classes.

The testing set would be used after training to test the performance of the classifier on a variety of acoustic sources. It is preferable that the testing set contain enough independent sources that the results have some statistical significance. Results of testing on two propeller planes, for example, cannot necessarily be used to describe the behavior of the system on all propeller planes. The testing set contained all available recordings from the data base for this study. Recordings from both the training and evaluation sets were included in the testing set. Ideally, the testing set would be different from the training and evaluation sets to give a fair analysis of the system's ability to recognize real sources. The limited number of recordings required that all be included in testing to give some statistical significance to the results. Differences in the signal processing between the training and testing sets resulted in different segments used for each, so the testing set was not a carbon copy of sources used in training. These differences are discussed in the section on signal processing.

The content of the different sets is too complicated to include in the text, so they are given in the appendix, by source along with a description of each source and each training set. General guidelines used to generate these sets are described below.

The jets could be easily be split up into training, evaluation, and testing sets due to the large number of recordings. For example, referring to Table 1, the 6 recordings of the 737 could be split up in pairs; two runs in the training set, two in the evaluation set, and all six in the testing set. This meant the testing set would contain 4 sources the system had heard, but two sources the system had never heard.

The propeller planes were split up differently to test the robustness of the classification system. The availability of both takeoff and landing recordings meant the system could be trained with takeoff recordings then tested with landing recordings. This would be useful for testing if a cluster generated with the one type of operation of a source could be used to correctly classify the same sources operating in different conditions. The training and evaluation sets that contained propeller planes contained recordings of those planes on takeoff only, while the testing set contained recordings of takeoff and landing.

The helicopter, wind turbine, and train were more difficult to split into different sets. The few recordings of these three sources were all of long duration, so they were split into segments for use in the different sets.

The helicopter recordings were split into nine segments. These nine segments could be split into training and evaluation sets, but the testing set would have to use the same recordings to test the results. As mentioned above, differences in the signal processing between the training and testing steps insured there was reasonable variation between these sets. The exact differences are described in the section on signal processing.

In a similar fashion, the wind turbine recordings were split into twelve segments and the train recordings into eleven segments. Again, this was enough to split into training and evaluation sets, but not enough for a totally independent testing set.

The exact content of a training set depended on the sources to be classified. For example, referring to Fig. 11, the training set used to generate the decision surfaces for distinguishing helicopters from other aircraft contained only helicopters and aircraft. The particular sets used in this study contained a large number of plane recordings because there were a large number of aircraft recordings available but only one helicopter. The classifier had to distingush a large number of planes, both propeller and jet powered, from the single helicopter used in the study. A cluster

representing all the planes required many planes in the training set, while the single helicopter required only a few recordings to generate a representative cluster.

The results of such a training set would be useful only for distinguishing those planes from that single helicopter. If reasonable results are obtained, then the classifier can be considered successful at separating that helicopter from those aircraft. A statement such as 'this classifier separates all helicopters from all aircraft' could not be based on the results of this training set because the training set did not contain examples of all types of helicopters.

4.2 Calculated Features

The specification of features for the system to use in describing the patterns is an important step in the design of the system. The designer must use knowledge about the patterns to be identified to produce a list of useful features. The feature selector described in section 3.3 eliminates the need for the user to give a perfect set of features for the system. The feature selector chooses a good set of features from a large candidate list supplied by the user. The candidate list could contain anything possible that might have relavance to classification; the feature selector would take care of reducing that list.

The features for the recognition of acoustic sources were to be derived from information acquired from a single microphone. This presented problems in specifying features, because all features had to be calculated from a single sensor. In the medical diagnosis system described in the introduction, measurements could be taken using a thermometer, white blood cell counter, etc. Several different sensors providing uncorrelated information could be used to describe the state of the patient. However, the acoustic identification system involved a single sensor with a single output; a time

varying voltage. This meant that all the features would be correlated to some degree, having been derived from the same time history.

The purpose of features is to impart recognition information, and an ideal feature set is one in which each feature contains information not contained in the other features. Correlated features cannot convey information totally independent of each other because they are correlated, so the use of a single sensor limits the effectiveness of the optimal feature set. Nevertheless, a single microphone can convey a substantial amount of information useful for recognition, as every person who has identified the sound of an aircraft can testify. This information must be extracted to insure the features convey the characteristics of acoustic sources.

The acoustic signatures of aerospace vehicles are produced by a combination of engine noise, body noise, and their interaction with the environment. These different noises are all due to a resonant structure excited in a periodic or semi-periodic manner, which makes the frequency distribution of energy a good descriptor of source characteristics as it is affected by structural size and geometry [17]. For example, some aerospace sources use engines and blades rotating at lower frequencies than other sources, hence their acoustic energy is likely to be distributed in comparatively lower frequencies. The distribution of spectral energy has been used before to calculate feature values to describe acoustic sources [1,2,4,17]. Table 3 shows a list of features used previously to describe the acoustic emissions of reactor feeder pipes [4]. This list illustrates the range of features that can be calculated from the output of a single sensor, using both time and frequency domain information.

These features were used as a basis for the features generated in this study, and were adapted to describe the characteristics of aerospace acoustic sources.

The characteristics of the sources to be identified are shown in the Power Spectral Densities (PSD) in Figs. 12-16. These PSDs are merely representative of those used in the study. Useful features can be described by comparing the sources.

Table 3. Features Used on Acoustic Emissions of Reactor Feeder Pipes [4]

Feature #	Description
1	standard deviation of AE signal
2	skewness coefficient of AE signal
3	kurtosis coefficient of AE signal
4	coefficient of variation of AE signal
5	rise time for biggest pulse in time domain
6	fall time for biggest pulse in time domain
7	pulse duration for the biggest pulse in time domain
8	pulse width for the biggest pulse in time domain
9	rise time for the 2nd biggest pulse in time domain
10	fall time for the 2nd biggest pulse in time domain
11	pulse duration for the 2nd biggest pulse in time domain
12	pulse width for the 2nd biggest pulse in time domain
13	pulse ratio of the two biggest pulses
14	pulse interval of the two biggest pulses
15	partial power in the frequency band 0 - 0.25 MHz
16	partial power in the frequency band 0.25 - 0.50 MHz
17	partial power in the frequency band 0.50 - 0.75 MHz
18	partial power in the frequency band 0.75 - 1.00 MHz
19	partial power in the frequency band 1.00 - 1.25 MHz
20	partial power in the frequency band 1.25 - 1.50 MHz
21	partial power in the frequency band 1.50 - 1.75 MHz
22	partial power in the frequency band 1.75 - 2.00 MHz
23	ratio of the 2 biggest partial powers
24	ratio of the smallest and biggest partial powers
25	frequency of largest peak in power spectrum
26	amplitude of largest peak in power spectrum
27	frequency at which 25 percent of accumulated power was observed
28	frequency at which 50 percent of accumulated power was observed
29	frequency at which 75 percent of accumulated power was observed
30	number of peaks exceeding preset threshold in power spectrum

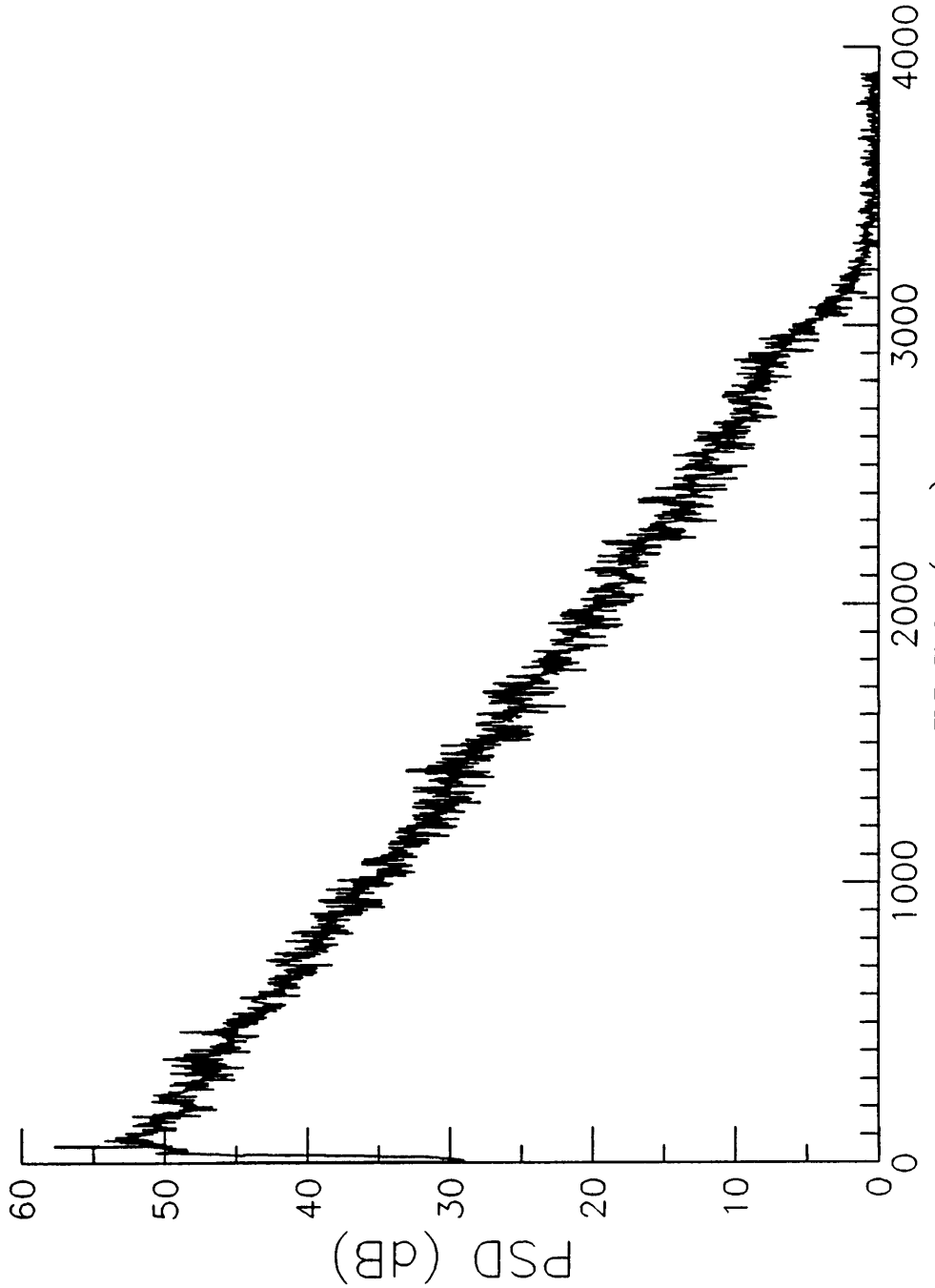

Figure 12. Power Spectral Density - Jet Plane

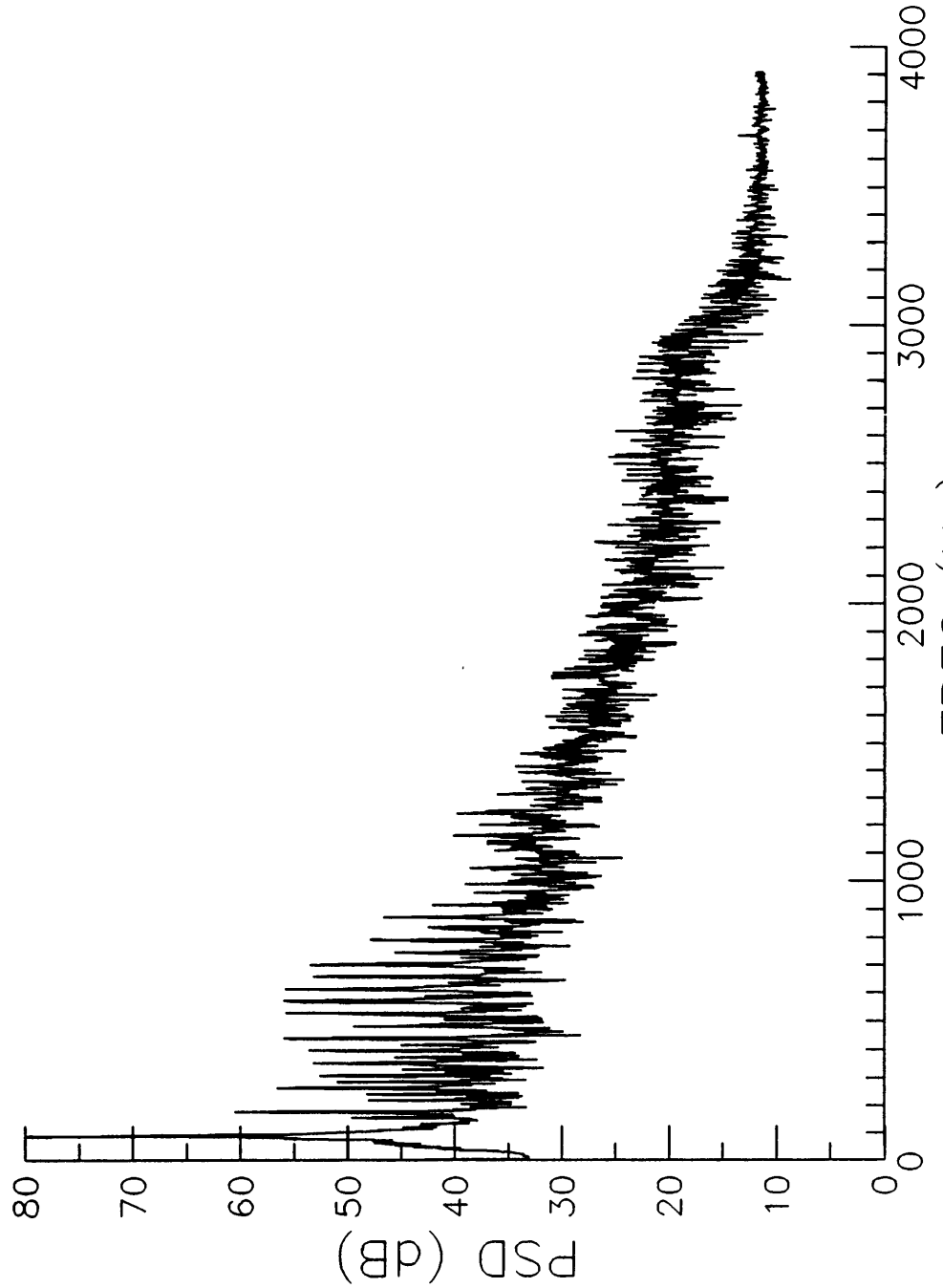

Figure 13. Power Spectral Density - Propeller Plane

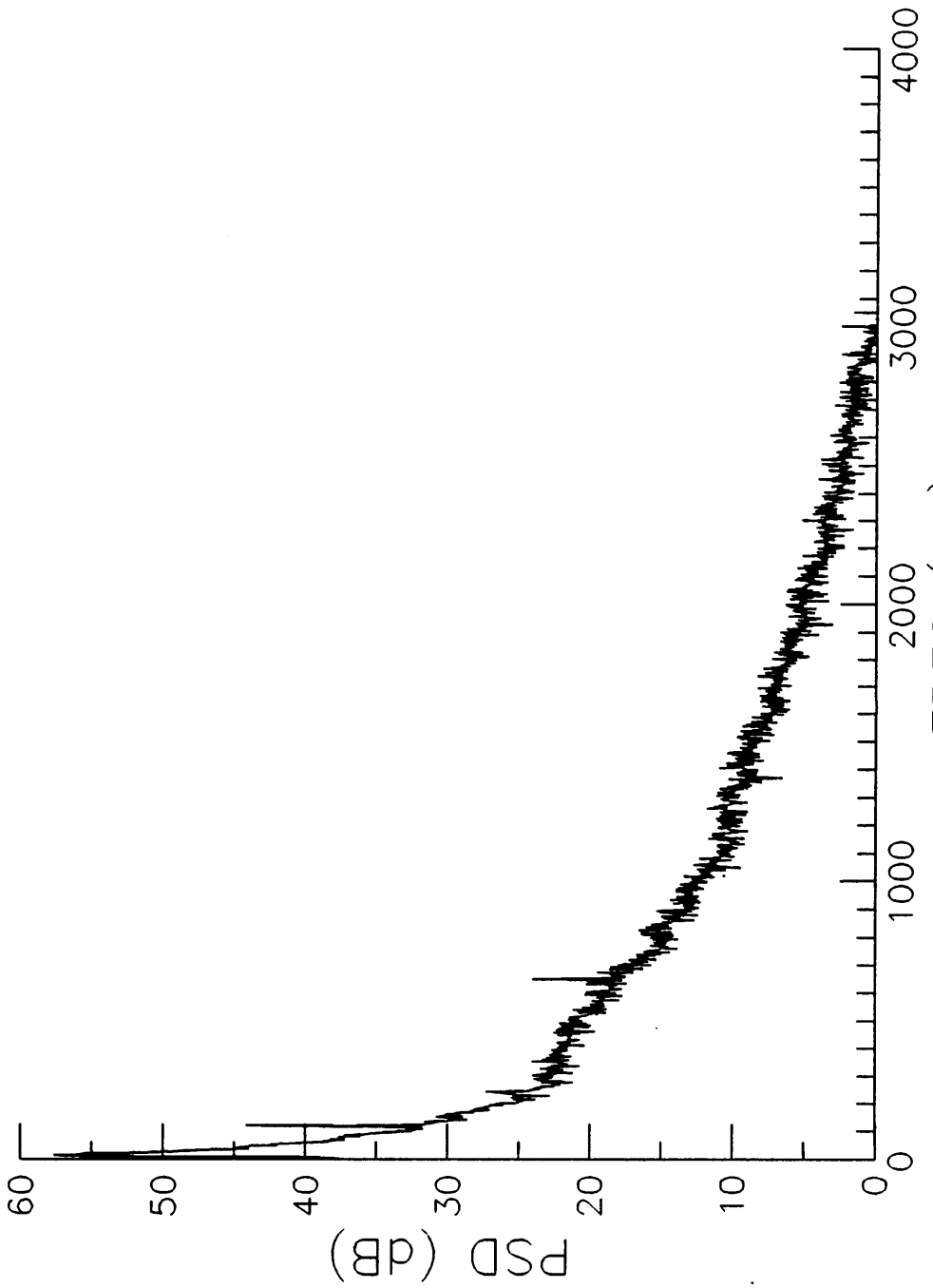

Figure 14. Power Spectral Density - Wind Turbine

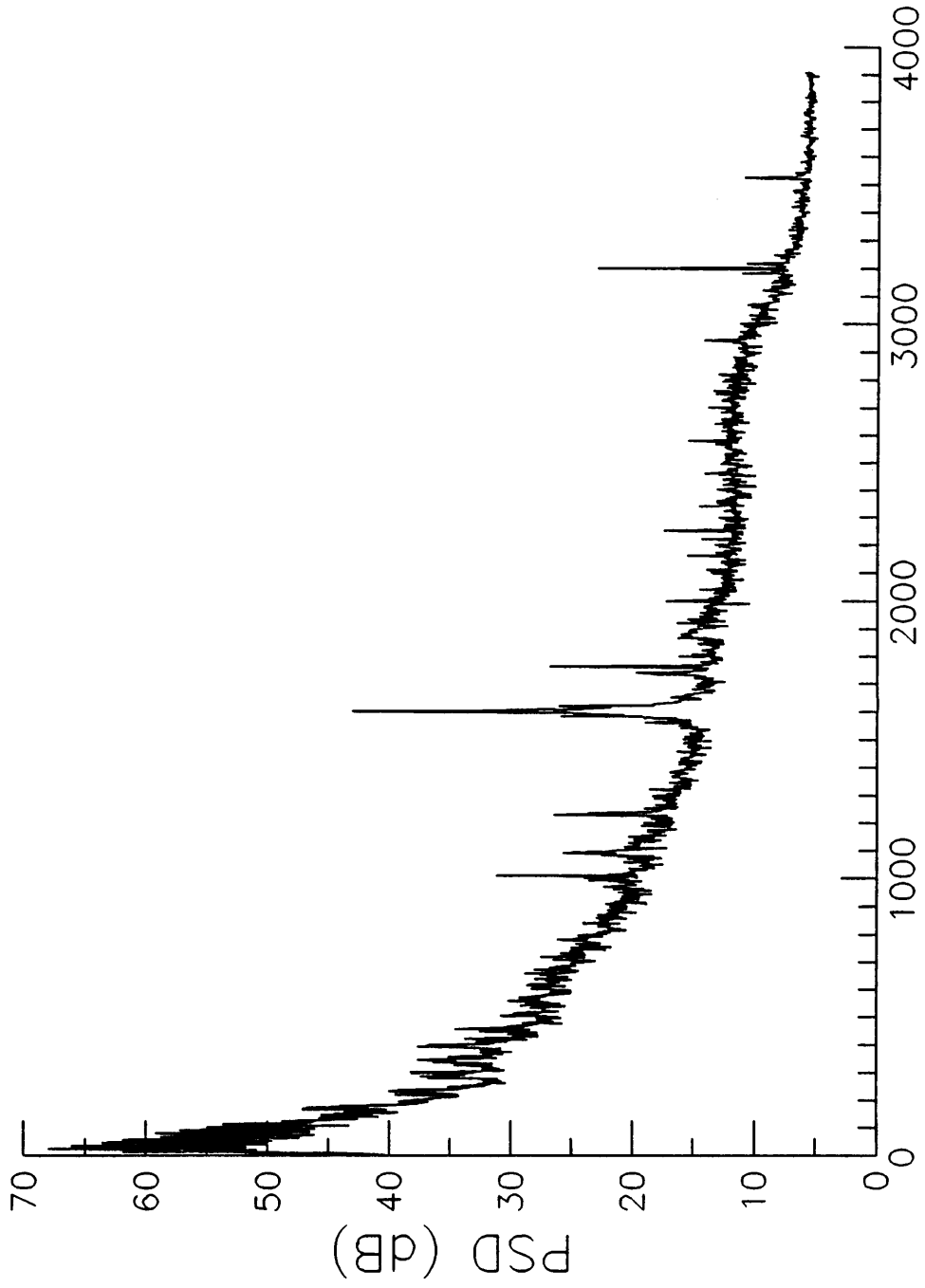

Figure 15. Power Spectral Density - Helicopter

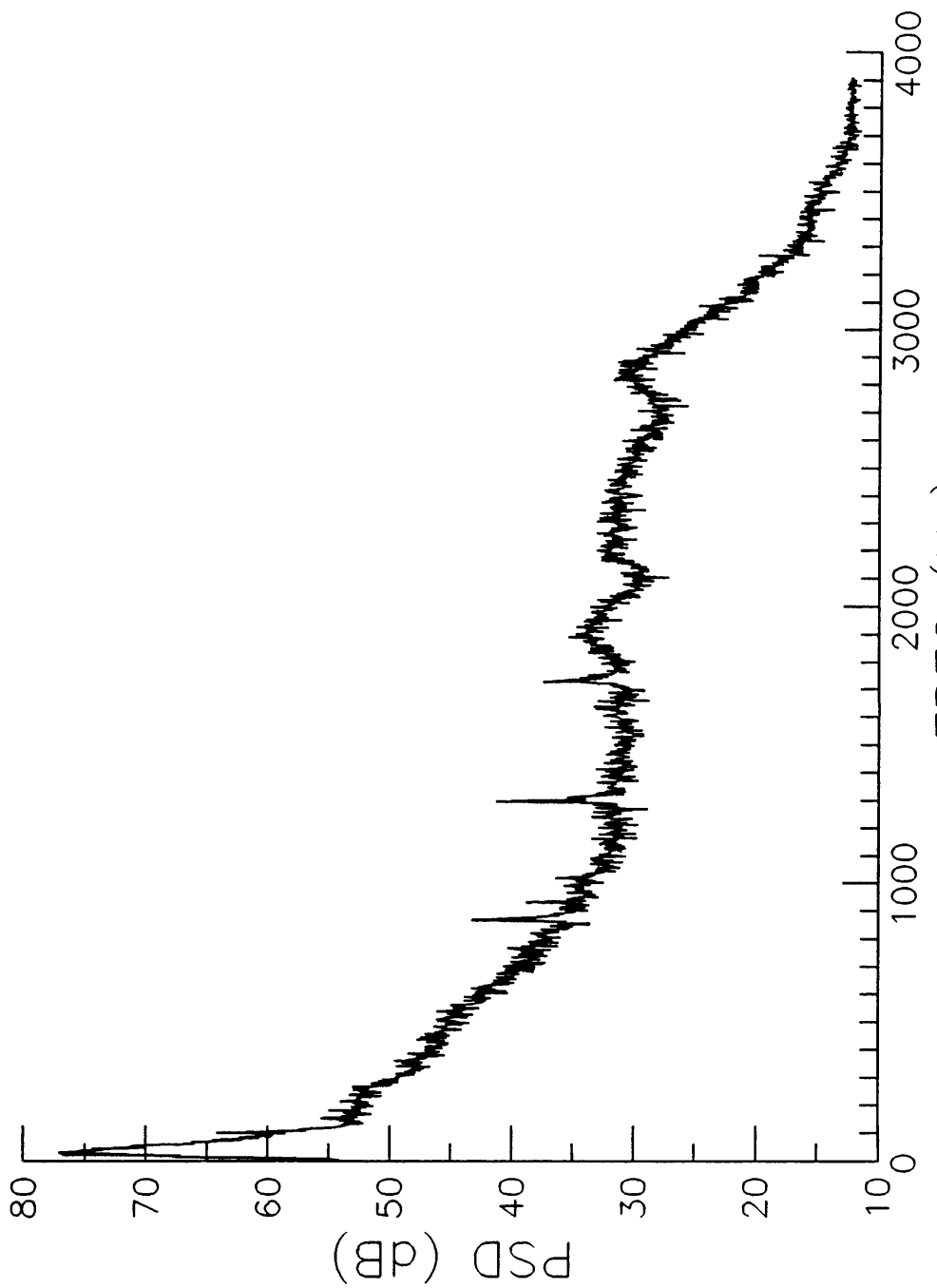

Figure 16. Power Spectral Density - Train

The PSD of the jet, Fig. 12, shows how the energy of the jet is spread throughout the spectrum. The PSD decreases at a much slower rate than the PSDs of the other sources. There are no peaks or valleys as frequency changes in the spectrum of the jet. In contrast, the PSD of the wind turbine, in Fig. 14, shows how its energy is concentrated almost entirely in the very low range, dying off to nothing with increasing frequency. There are a few small peaks, but most of the energy is at the very bottom of the frequency range.

The PSDs of the propeller plane, helicopter, and train, Figs. 13, 15, and 16, show these sources also have a substantial portion of energy in the low frequency range. The differences between the sources suggest a feature that conveys information about the distribution of energy throughout the spectrum, such as the frequency location of 75 percent of the spectrum's accumulated energy. This frequency would be higher for the jet than the others, since the jet's energy is spread more uniformly throughout the range, and lower for the wind turbine, since so much of its energy is in the very low range.

The PSD of the helicopter also has some unique characteristics. The helicopter differs from the other sources in that it has a large peak in the 1000 to 2000 Hz range. This peak contains a large amount of energy, relative to the areas around it, so a useful feature would be a description of the distribution of energy in the 1000 - 2000 Hz band. The distribution of energy in this band for the other sources is more constant, so this feature will differentiate the helicopter.

Numerous other features can be imagined to describe these sources, but certain features are undesirable. The signatures of these sources are subject to environmental conditions, such as ground and building reflections, changing weather condition, and Doppler shifts. Features that are affected by such conditions can confound the recognition system.

These effects produce small changes in a source's spectrum that can reduce the utility of certain features. A Doppler shift can cause a propeller tone to change in frequency as the plane flies over, so a feature describing the frequency of a propeller tone will be a poor descriptor of that source.

According to that feature, the source will have a characteristic of a propeller plane on approach, but receding it will not have that characteristic. A feature such as this is undesirable. A feature that describes the number of peaks above a certain threshold could be useful, because the number of peaks is unaffected by shifts in frequency. The propeller plane in Fig. 13 would be well described by such a feature, since there is large number of tones under 1000 Hz, compared with the other sources.

Features that describe the location of energy within a band are less sensitive to shifts such as the one discussed above, if the bandwidth of the feature is not so narrow as to be centered around one peak. If the feature describes the amount of energy produced by a certain tone at a certain frequency, then the energy will change as the source moves by so the feature will be useless. The bandwidth would have to widened so that even if the tone shifts, the energy will still be present in the band and the feature will still contain useful information.

Another consideration in the specification of features is that sources will have different levels of loudness, depending on their location relative to the microphone. Loudness would rarely contribute any information as to a source's identity, so the feature values are normalized to eliminate these effects. A feature that describes energy in a bandwidth can be normalized to express the percent of total energy in that bandwidth, reducing the effects of loudness.

Continuing in a manner similar to that described above, 108 features having some relevance to classification were devised for the present study. These 108 features are listed in the appendix. All the features were normalized as percentages, so their values ranged from 0 to 100.

The features were devised in such a way that they could be applied to any range of the spectrum, and by varying a few parameters, different features could be calculated. For example, one set of features required the calculation of the energy in a given overall bandwidth, say 1000-2000 Hz, which was then partitioned into smaller bandwidths. In this example, those smaller bandwidths might have been 1000-1100 Hz, 1100-1200 Hz, etc. up to 1900-2000 Hz. Different features were

then used to specify what percentage of energy in the overall bandwidth was present in each of the smaller bandwidths.

Features calculated in this manner could be easily specified and easily changed to accomodate different bandwidths. Notice that the above example could be changed by specifying a different overall bandwidth and different smaller bandwidths, but the calculations still use the same algorithm; first calculate energy in the overall band, then express the percentage of that energy present in the smaller bands.

Another repetitive feature type used was the accumulated power. These features gave the frequencies at which some percentage of the total energy in a band occured when accumulated up from the bottom of the band. If a bandwith of 1000-2000 Hz was being used, and 50 percent of the accumulated power occured at 1250 Hz, then a feature used to designate the 50 percent accumulated power point took on the value 1250 Hz. Actually, due to the normalized values, the feature was assigned the value (1250-1000)/(2000-1000) = 0.25.

Ratios of powers in different bands were also used. For example, after the calculation of percent powers in the smaller bandwidths described above, a ratio could then be formed comparing the amount of power in the 1200-1300 Hz band to that in the 1800-1900 Hz band. These features conveyed slope information about the PSDs.

The table of calculated features in the appendix shows the periodic nature of the feature calculation. A whole group of features could be calculated by specifying the bandwidth of interest, the number of smaller bandwidths within the overall band, the accumulated powers points, and the ratios to calculate. Once these features had been calculated, the whole process could be started over with a different bandwidth, number of smaller bandwidths, etc. This is evident in the list of features in the appendix.

Of the 108 features, only two were not calculated from the PSD, those two being calculated from the autocorrelation. No features were calculated straight from the time domain.

The time domain features were affected by the motion of the airplanes as they passed over the microphone. The amplitude of the plane's signature changed as it approached the microphone, so features taken straight from the time domain were not useful. A feature such as number 9 in Table 3, the fall time for the second biggest pulse in the time domain, is meaningless when the time domain signal is changing so rapidly.

Recognition is not determined solely on the presence or absence of a single feature, however. Recall that the patterns are described by an n -dimensional feature vector, so all the features play a part in recognition. For example, a pattern that has small peaks in the 1000 to 2000 Hz band and a large amount of low frequency energy would be a train, while one with large peaks in the same range and a lot of low frequency energy would be a helicopter (see Figs. 15 and 16). The features have magnitudes, not just on-off characteristics, and several features together contribute evidence as to a source's identity.

4.3 Signal Processing

The candidate features required the calculation of the PSD and the autocorrelation. This required the digitization of the recordings so the computer could calculate these quantities. This section describes those aspects of the acquisition of the signals related to signal processing. The range of the analysis, sampling rate, filtering, and segmentation of data are discussed.

4.3.1 Processing Range

The range of interest affected the sampling rate, number of points, filtering, and also the information available for feature calculation. The sampling rate should be at least twice the highest frequency of interest, the number of points should be chosen to give a desired frequency resolution, and a low pass filter needs to be used to reduce aliasing. In addition to these considerations, the range had an affect on the information useful for classification.

The acoustic signatures of the sources under consideration contained information useful for identification below 5 kHz [1]. The wind turbine and train had considerable information in the very low frequencies. These sources contained little characteristic noise in frequencies above 1000 Hz. On the other hand, the jet had energy throughout the spectrum and the helicopter had a significant peak in the 1000 to 2000 Hz range.

A range that was useful for the wind turbine and train, such as 1000 Hz and below, could leave out information important to the identification of the jet and helicopter. On the other hand, a large range that contained all the information necessary for the jet and helicopter, for example 10000 Hz and below, could compress the low frequency information characteristic of the wind turbine into too small a range.

The needs of the different sources were compromised by selecting a range of 2.5 kHz. There was still information about the helicopter and jet in this range, but the wind turbine's energy wasn't totally hidden at the very low frequency end.

4.3.2 Sampling Rate and Frequency Resolution

The next consideration was the sampling rate and the frequency resolution. The resolution is determined by both the sampling rate and the number of points processed. Resolution is important for the precise location of tones in the PSD, such as the exact frequencies at which propeller harmonics occur, but the features that were to be calculated did not rely on the precise location of these tones.

The data acquisition system that was to be used had preset sampling rates, so a rate of 7812.5 Hz was selected. This rate was well above two times the highest frequency of interest, 2.5 kHz.

The number of points processed determines how sharp the frequency resolution. A large number of points gives the PSD very fine frequency resolution, while the PSD has less resolution if a small number of points is used.

In addition to the resolution of the PSD, the number of points affected the time required for sampling; the more points, the longer the required sampling time. A long sampling time becomes a problem when the signals being sampled are changing with time and cannot be considered stationary. Most signal processing algorithms assume a signal is stationary for simplicity, so if a large number of points is used for sampling, there is a danger that the PSD and autocorrelation will not accurately reflect the properties of that segment. The airplane recordings were the most nonstationary, since their properties changed as the planes flew over the microphone.

The number of points for this application was chosen to be 4096, which gives a segment 4096*1/7812.5 seconds in duration, or 0.52 seconds. The airplane recordings could be considered stationary for this period of time if they were sampled on approach, and the period of closest flyover was avoided. At the closest point of flyover, the signal was changing the greatest relative to the microphone, so this time was to be avoided.

The points of data were not windowed in any way. The recordings were split into consecutive segments of 4096 points and then processed. A window would reduce the energy from one frequency leaking onto neighboring frequencies, thereby improving frequency resolution. None of the features calculated required the amount of energy found in a single tone, nor required the precise location of tones in the PSD, so a window was not used.

4.3.3 Processing of Training Data

The generation of training and evaluation sets of data required the digitization of the source recordings. The digitized recordings were used to calculate the PSD and autocorrelation, from which the features could be calculated to form a pattern's feature vector. The feature vectors for the patterns then had to be put into an appropriate training or evaluation set for use by the system to select features and calculate the decision functions.

The components of the acquisition system used for the training patterns are shown in Fig. 17. The Nagra IV-SJ tape recorder was used to play the tapes. The output of the Nagra went through a Tektronix oscilloscope so the level of the signal could be monitored during playback. The output then went to a Precision Filters Model 6000-C-1-C system, where 20 dB of pregain was added, and the signal was low and high pass filtered. The low pass filter was set at 3.1 kHz, and the high pass was set at 1 Hz. The low pass filter was set to reduce the aliasing of energy from frequencies more than half the sampling rate. Both filters had a rolloff of at least 40 dB/octave.

The signal was sampled by transient data recorders, Pacific Instruments model 9830, which stored 524288 points of the time history. These points were then transferred to a Microvax workstation using a standard IEEE interface.

Data acquisition began by starting the Nagra tape recorder, and when the signal was visible on the oscilloscope, the transient data recorders were activated, storing 524288 points of the time history.

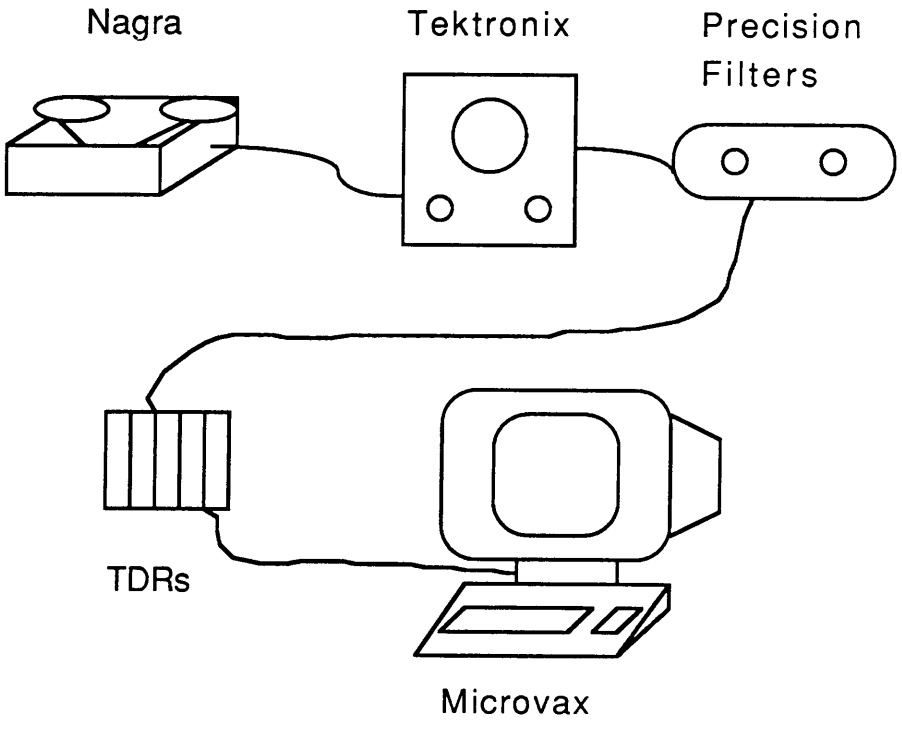

Figure 17. Data Acquisition System - Training

These points were transferred to the Microvax, and the system was ready for the next source recording. This was continued until all the sources had been digitized.

The 524288 points sampled by the transient data recorders corresponded to 67.1 seconds of recorded signal. The wind turbine and train recordings lasted approximately 60 seconds, and were much more stationary than the plane recordings, so the digitized time histories of these sources were split into two or three pieces to provide more runs for training. This partitioning is shown in Fig. 18. The complete run is shown at the top of the figure, and the pieces are shown below the complete run. The length of the pieces varied from run to run and piece to piece.

The airplane recordings of takeoffs and flyovers all lasted considerably shorter than 67.1 seconds. For these recordings, the pieces corresponding to the approach of the aircraft were used for processing, and the rest was ignored.

Once all the processed sources were on the Microvax, features were calculated from the PSD and the autocorrelation. The PSD and autocorrelation were calculated using the 4096 point sample segment. After a recording had been partitioned, as described above, the PSD and autocorrelation of each of the remaining 4096-point segments were averaged together to reduce random noise in the spectrum. The feature values for each source were put into a vector for that source, and the training sets were constructed of appropriate sets of these source vectors. The training and evaluation sets are listed in the appendix.

These sets were then read in by the classification system and were used to select features and calculate the decision functions. The program had to be run a separate time for each classifier using the training and evaluation sets constructed for that classifier; the ground versus aerospace classifer, the train versus wind turbine classifier, etc. Each classifier required a run of the feature selection routine and generation of decision functions. The results of each training run were stored for later use in classification. The results included a list of the best features selected from the 108, and the

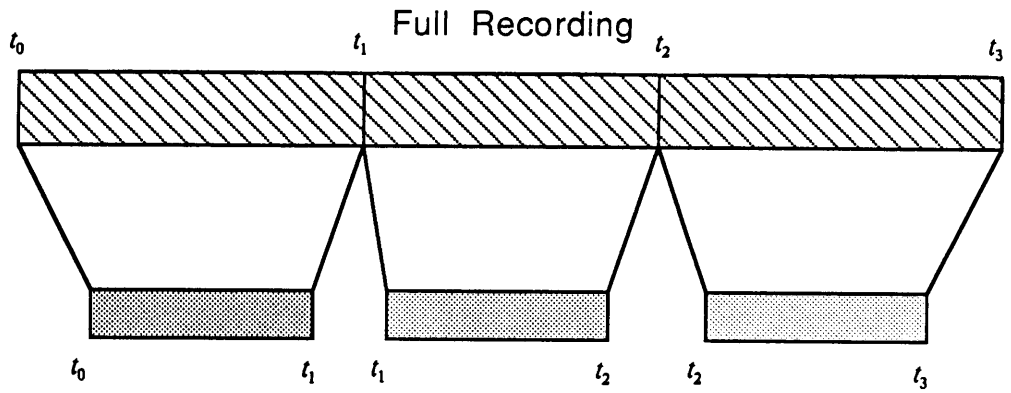

Figure 18. Data Partitioning - Training and Evaluation Sets

coefficients of the weight vectors of the decision functions. A set of results was generated for each classifier.

Feature selection was very time consuming since it involved the evaluation of hundreds of different combinations of features, each of which required the construction and testing of a classifier, as described in section 3.3. This selection was required only once for a given problem, though, because once the features had been selected and the classifier trained, the results were used in a simple multiplication of the weight vectors of the decision functions.

Computer listings of the codes used for feature calculation, feature selection, and classifier training are given in the appendix.

4.3.4 Processing of Testing Data

The results of training and feature selection were to be tested using sources similar to those used for training. Testing was meant to be real time, so that a time history was acquired, processed, features calculated, and the resultant pattern vector classified using the results of training. The testing sources came from the same data base as the training and evaluation sets, but were processed differently.

The acquisition system for the testing phase is shown in Fig. 19. The same Nagra tape recorder and set of tapes were used as a source of the source recordings, but the physical acquisition system was different.

For testing, the recorder output was monitored with headphones, eliminating the oscilloscope used in training. Data acquisition was begun when the signal was audible in the headphones. The signal then went to a Precision Filters Model 216, where pregain and filtering was done as described in the training mode above. A Data Translation board, DT 3362, sampled the signal and the time

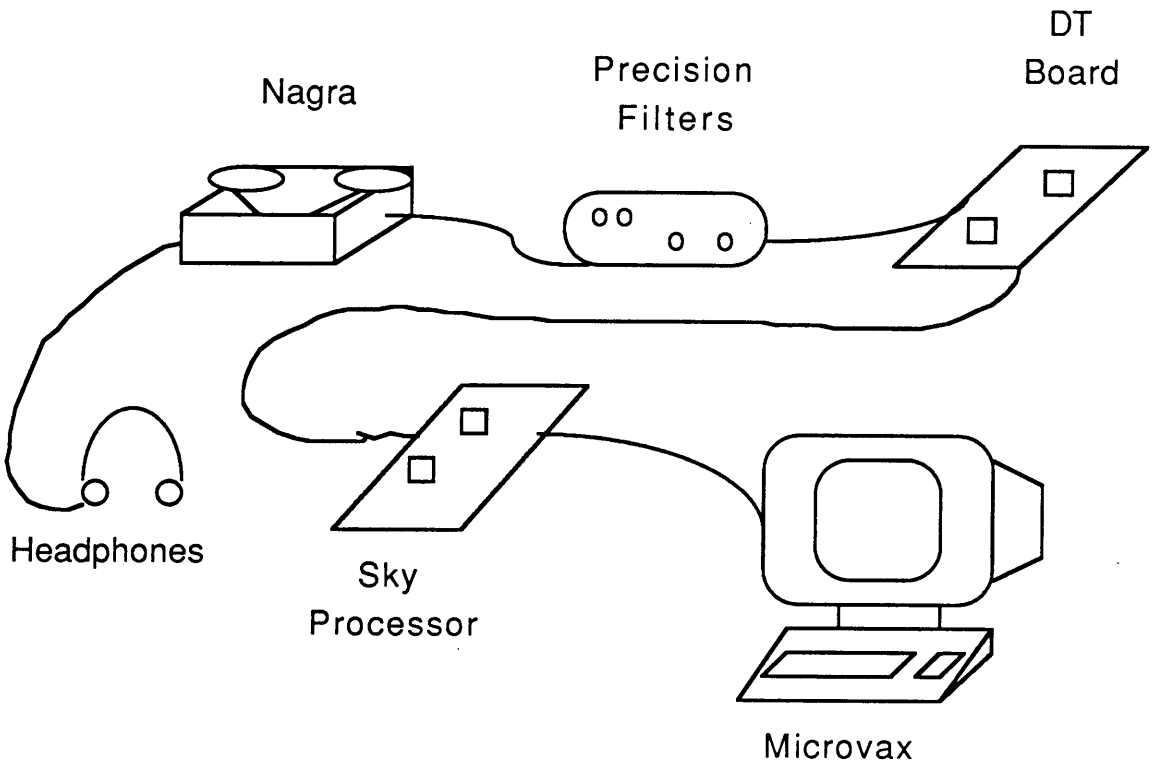

Figure 19. Data Acquisition System - Testing

history was stored by a Sky 320 processor with 64k memory. Again, the time histories were sent to a Microvax using a standard IEEE interface.

All points of the time history sampled by the testing system were used for processing; none were thrown away as in training. The testing system sampled only 64k points, which corresponded to 8.4 seconds of data. This was short enough to contain approach sounds of all of the aircraft so all 64k data points were used. The relatively small number of sampled points allowed the partitioning of the longer recordings into many shorter pieces, in effect creating a greater number of test sources. This is shown in Fig. 20. Partitioning was done only on the train, wind turbine, and helicopter recordings. Each recording had substantial variation from one section to the next, so even though the pieces came from the same source recording, there was variation between the pieces.

The partitions also introduced variation to the recordings used for both training and testing. A comparison of Figs. 18 and 20 illustrates this point. Assuming the recording is the same in both figures, the pieces used for training and testing come from different, albeit overlapping, portions of the recording. While both may have come from the same recording, they were not identical pieces and hence there was some variation between the two sets.

Processing of the airplane recordings used in testing also introduced variation from the same recordings used for training. Both training and testing required the triggering of the acquisition system, and in both cases triggering was done manually, either on a visual or audible cue. There is very little chance that a recording used in testing was triggered at the same point as when the recording was used in training, which introduced small variations between the training and testing sources.

Differences between the training and testing sources are described to emphasize the system was not reclassifying the same digitized segments used in training. Instead, there was variation due to the different segments of the recordings used for the two steps. For the jets, there were enough

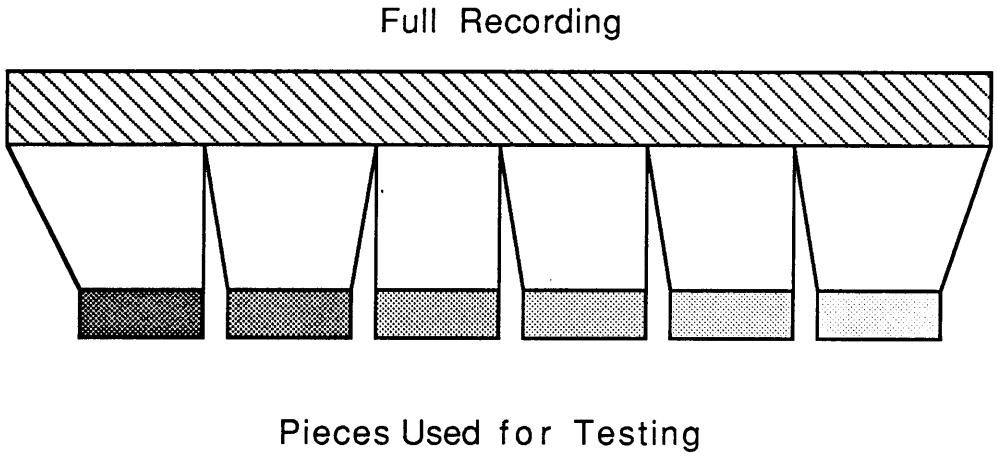

Figure 20. Data Partitioning - Testing Set

recordings that a substantial number that were used for testing were not used in any way for training.

After the time history was transferred to the Microvax, all processing was done immediately. This included the calculation of the PSD, autocorrelation, features, extraction of selected features for each classifier, and evaluation of stored decision functions. The calculation of the PSD and autocorrelation is described in detail elsewhere [20,21].

The details of the classification and testing are given in Chapter 5, along with a discussion of the results

5.0 Results

This section describes the ability of the trained system to identify test sources. For testing, the system was presented with a source recording, the recording was processed as described in the preceding chapter, and the resultant feature vector was classified as belonging to one of the pretrained classes.

Chapters 2 and 3 described the method of representing a pattern as an n -dimensional pattern vector containing the values of the features. This pattern vector was classified using decision functions of the form $d(\overline{x}) = \overline{w}'\overline{x}$, where $\overline{x}$ represents the pattern vector and $\overline{w}'$ represents the weight vector of the decision function, determined in the training mode. One decision function was used for each of the classes under consideration. The pattern $\overline{x}$ was classified by evaluating each decision function and assigning it to the class of the highest valued decision function.

The system's decision on the identity of the source was recorded and compared with the actual class of the source. This was done for every test source so a count of errors and correct classifications was generated. The recognition rate was calculated as the percent of the test patterns correctly identified by the system.

In the following discussion, the term *classifier* is used to refer to a single run with a single number of features, such as the *classifier* of run 3 with 6 features. The term *classification system* is used in a more general sense to refer to the different classifiers as a group.

5.1 Description of Test Runs

The purpose of this investigation was to evaluate not only the ability of the classifer to identify unknown sources, but also to evaluate the performance of different classifiers. The different classifiers are briefly discussed below and in more detail in Chapters 3 and 4.

Chapter 3 described 2 different architectures to be tested; a tree structure, and a single level structure. The tree structure made several intermediate classifications before a pattern was fully identified, while the single level structure immediately classified a pattern into a source category without intermediate classifications.

In addition to testing two different types of structures, two different types of features were to be tested. These two types were based on two different ways of calculating the energy in bandwidths from the PSD; summing the dB units linearly and summing them non-linearly.

The energy calculations required a summing of the PSD values as part of the calculation. One method summed the PSD values using the normal non-linear method of summing dB units, in which ten is raised to each value divided by ten, then summed. This gives the true energy in the spectrum for the feature values, and the features calculated using this method are henceforth called regular features. Unfortunately, this method is subject to the very large range of values afforded by the dB units. Energy in a large lower frequency peak tends to dominate the energy in the

spectrum. The PSD plots shown in Chapter 4 hide this fact because the dB units compress the actual range of values.

The second method of calculating the features summed the dB PSD values linearly, disregarding the dB properties. The features calculated in this manner are henceforth referred to as pseudo features. This method was less sensitive to large peaks in the low frequency end dominating all the energy, and was really just comparing the shapes of the PSDs for identification.

The wind turbine's spectrum in Fig. 14 illustrates the differences between the two types of features. The large peak below 300 Hz contains a significant amount of energy, relative to the higher frequencies. The regular features would convey this actuality; a bandwith below 300 Hz might have 95 percent or more of the energy present below 2000 Hz. Features that described the energy present in the bands up to 2000 Hz would have small values, less than 1 percent in some cases. This reduces their utility in imparting information since any changes in the spectrum at the high end get lost in the significant digits. The pseudo features, on the other hand, might calculate that only 75 percent of the energy below 2000 Hz is present in the band below 300 Hz. Features describing the energy in the bands above 300 Hz would have larger values, enabling them to be more descriptive of the spectrum.

The pseudo features do have their drawbacks, however. Just as they are less sensitive to large peaks in the low frequency end, they are less sensitive to peaks throughout the spectrum. The effect this will have on recognition ability can be verified best through testing, and the hypothesis that the pseudo features can convey more recognition information was to be tested by comparing the performance of the two different types of features.

The different structures and different features resulted in four test runs, labeled below. In subsequent discussions of the results, the runs are referred to by their run number only.

1. Run 1 - Tree structure, pseudo features

2. Run 2 - Tree structure, regular features

3. Run 3 - Single level structure, pseudo features

4. Run 4 - Single level structure, regular features

The classifiers for each run were constructed using 4-10 features to test the feature selection algorithm and evaluate system performance with a varying number of features, The selection process was run for each classifier. For example, once for the ground versus aerospace classifier to select four good features, then run again to select five features, and so on through 10 features. This was done for every classifier and every run.

The selection of so vast a number of features and their use in the pattern vectors made interpretation of their effects very difficult. The tree structure required training and feature selection for four classifiers; ground vs. aerospace, train vs. wind turbine, helicopter vs. fixed-wing aircraft, and jet plane vs. propeller plane. These were trained for runs 1 and 2, and in addition, two single level classifiers were trained for runs 3 and 4, making ten total classifiers. Each classifier was trained using 4-10 features, meaning the feature selector was run 70 different times producing 70 different sets of optimal features dependent on the sources to be classified. These 70 sets of features were used in vectors that were multiplied in a weighted fashion to produce a scalar output. The effect of each feature in a given decision varied according to the coefficients in the weight vector of a given decision function. The weight vector for one decision function may have multiplied a feature by 100, while another may have multiplied it by 0.5. Determining the relevance of a given feature to classification was difficult, especially when there was more than one decision function. For this reason, the results were not interpreted in terms of the role each feature played in classification. The appendix contains a list of the features chosen for each classifier in this study.

5.2 Sources Used for Testing

The test sources were from the same data base as the training sources. The processing of the test sources was described in section 4.3.4, which detailed the partitioning of the longer recordings of the wind turbine, train, and helicopter. The takeoff and flyover recordings of the aircraft were not partitioned since they were not long enough to be split into pieces. All the recordings available in the data base were used for testing, resulting in approximately 170 test sources. This number included the pieces of the longer recordings, each considered as a separate test source. The exact number of test sources varied from run to run depending on occasional hardware problems that prevented the processing of a recording.

Table 4 shows the number of test sources that came from each of the different classes, for each of the runs. This table shows that over 50 percent of the test sources in any given run were jets.

From a statistical standpoint, the greater the number of test sources, the greater the confidence that the test results accurately represent the behavior of the system. The large number of jets produced numerical results that could be interpreted as representative of the system's behavior on that class with confidence. However, the small number of test sources from the other classes produced results on those classes that could only be viewed as indicative of the system's performance. The numerical results were useful for describing the performance of the system but should not be used in a number by number comparison.

The test recordings were not duplicates of those used in training, due to the differences in processing for the train, wind turbine, and helicopter, and due to different recordings for the planes.

The results for each run, tabulated in the appendix, contain a list of the sources used for testing and a brief description of each source.

Table 4. Test Sources

Type	Number
Jet Plane	92
Propeller Plane	29
Helicopter	14
Train	18
Wind Turbine	20

5.3 Operation of Classification System

A classification run began by triggering the data acquisition system when the signal from the tape recorder was audible. The system sampled 64k points from which the microvax calculated the PSD and autocorrelation. All 108 features were then calculated and assembled into a pattern vector describing the sampled source recording. The pattern vector was classified by retrieving the appropriate decision functions stored in memory. This was a one step process for runs 3 and 4 since the classifier was single level, but the mutli-level classifiers of runs 1 and 2 had to be called up based on the decision of the previous level.

Figure 11 illustrates this multi-level classification process. The first level of classification was to separate aerospace and ground sources. If the source was classified as aerospace, then the classifier used to distinguish helicopters from fixed-wing aircraft was retrieved. If the source was further classified as a fixed-wing aircraft, the propeller plane-jet plane classifier was retrieved. Each of these classifiers had its own optimal features and weight vectors for the decision functions. Classification ended when the source fell into one of the end categories.

The final result of both the multi-level and single level classifiers for any one source was the assignment of the source recording into one of the five pretrained classes.

The same pattern vector was classified by each classifier, 4-10 features, for each run. A pattern vector was read in, it was identified with the 4 feature classifer, then the 5 feature classifer, and so on, until the 10 feature classifier had identified the same pattern vector. This allowed a direct comparison of the performance of the classifiers in a given run constructed with different numbers of features.

The same pattern vectors were not classified by the different runs. Run 1 was tested with the 170 recordings, then run 2, etc. The same recordings were used for each run but the exact same digitized

pieces were not used, so there was variation in the pattern vectors identified by the different runs. The effect of this variation was assumed to average out over all the test sources so that performance of the different runs could be compared with one another. The variations were small enough to assume that no classifier gained a benefit over the others by fortuitous sampling of a source recording. Each classifier was identifying helicopters or jets or trains, regardless of minute variations in the sampled segments.

The time required to classify a signal was recorded from start to finish to evaluate the feasibility of real time operation. The time required for a classification run was constant, since the acquisition time did not vary, and the same calculations were required for every source. This classification time is given in Table 5. The largest part of each run was the calculation of the PSD and autocorrelation, which made up 77 percent of the total time required. The classification of a pattern vector took only 1 second, even for the tree classification runs where multiple classifiers had to be retreived.

The times given in Table 5 indicate that real time operation of the system is feasible, especially if a dedicated FFT processor were used, eliminating the bottleneck in the PSD and autocorrelation computation.

5.4 Recognition Rates

This section gives the results of the classification system testing. These results are presented as the percentage of the test sources identified correctly, referred to as the recognition rate. For example, a 90 percent recognition rate indicates that 90 percent of the test sources were correctly identified by a classifier. The recognition rates were calculated from the presentation of 170 test sources described in Table 4, in four different runs, each trained with 4-10 features. The system's responses

Table 5. Classification Processing Times

Process	Time(sec)	Percent
Digitization	8.4	20
PSD and AC Calculation	33.1	77
Feature Calculation	0.5	1
Signal Classification	1.0	2
Total Time	43.0	100

for the test sources are given in the appendix and are summarized in this section. The results are given in terms of overall recognition rates, and then broken down by class of source.

5.4.1 Overall Recognition Rate

The overall recognition rate for each classifier is the percent of the 170 test sources identified correctly by the classifier. These rates are shown in Fig. 21 by run and number of features and are also tabulated in the appendix. The four different runs are identified with the symbols indicated in the key.

Each symbol gives the performance of a classifier with the indicated number of features. For example, a classifier from run 3 constructed using four features correctly identified 65 percent of the test sources, while a classifier from run 3 constructed using 5 features correctly identified 71 percent of the test sources. The lines between symbols do not indicate how performance varied between integer numbers of features, but are included to give a measure of continuity to the performance of each run.

Figure 21 shows that a classifier constructed with 8 regular features and using a tree structure, correctly identified 90 percent of the test patterns. This was the best recognition rate of all the classifiers. The lowest recognition rate was a classifier constructed with 4 pseudo features using a single level structure, which correctly identified only 65 percent of the test sources. A classifier from the same run correctly identified 85 percent of the sources with 8 and 10 features, so that run had some good classifiers. The recognition rates of the classifiers from runs 1 and 4 varied between 78 and 88 percent.

The number of features had no effect on the performance of the classifiers. There was no clear indication of one number of features having greater discriminatory power over another. Run 1 had a peak recognition rate with 6 and 8 features, run 2 with 8 features, run 3 with 8 and 10 features,

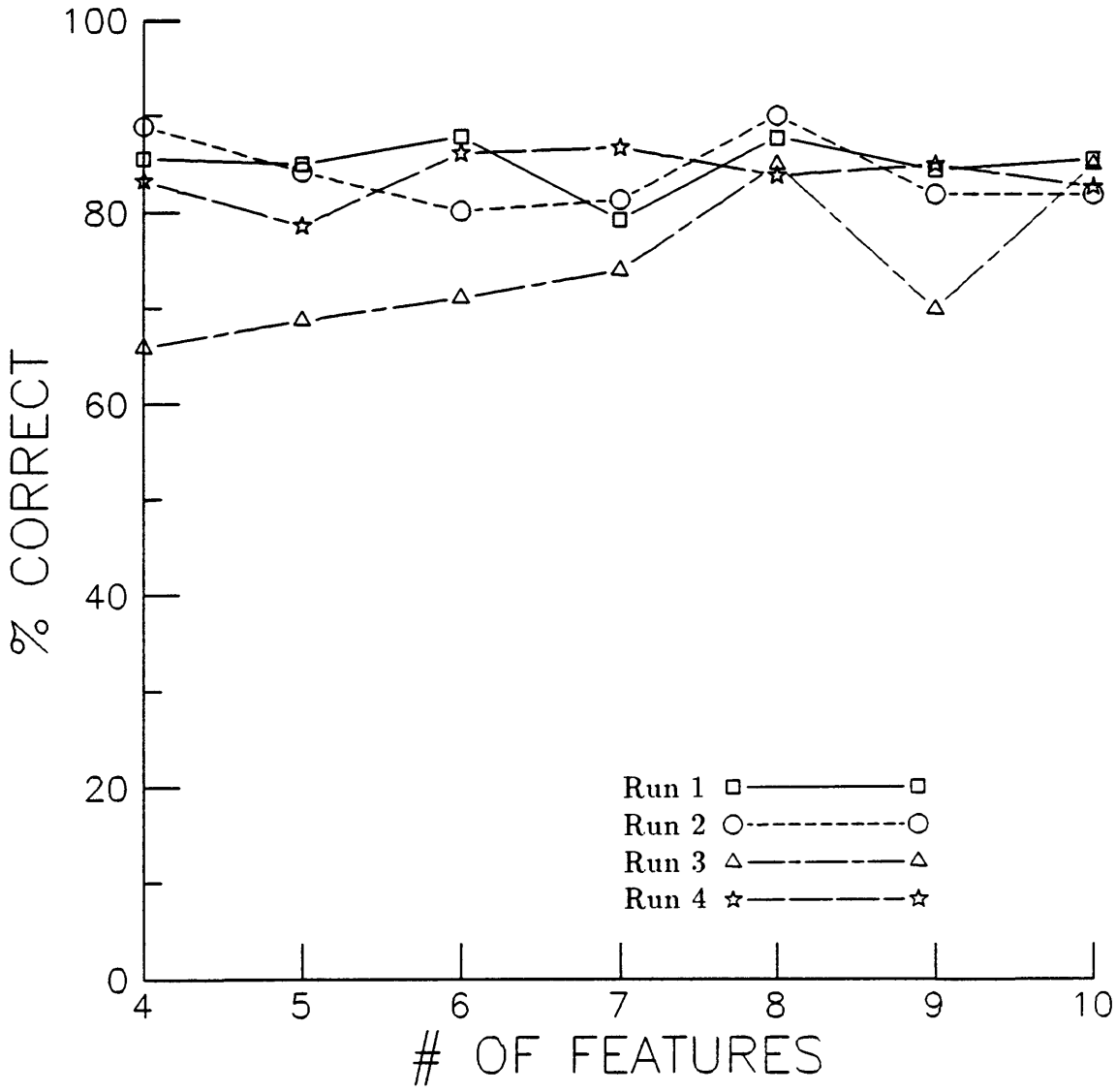

Figure 21. Total Recognition Rates for All Runs

and run 4 with 7 features. Runs 1,2 and 4 performed as well with 4 features as they did with 10 features. Run 3 did show some improvement with increasing number of features, until 9 features were used, which caused recognition to drop by 15 percent.

The next section details the performance of each run on each type of source. Since the total recognition rate was calculated by summing the errors made over the entire set, the performance by source is useful in pinpointing those sources that were particularly difficult for the classifiers to identify, or when poor performance on one type of source affected the overall performance of a classifier.

The recognition performance of each run on the different sources is given in Figs. 22-26. These plots are identical to Fig. 21, in that the different runs are identified by their respective symbols.

5.4.2 Jet Plane Recognition

The recognition rates shown in Fig. 22 for the jet planes are similar to the overall recognition rates in Fig. 21, which is to be expected since jet planes accounted for 53 percent of the test sources.

Runs 1 and 2, using tree classifiers and 4 features, correctly classified 95 percent of the jets. The jets included the nine different commercial airliners and the military jet, so the tree structure could successfully identify the class members even with the variation between different types of jets. The recognition rates dropped off slightly with 5-10 features, but stayed above 80 percent for runs 1 and 2. As with the overall recognition rate, the results did not highlight the presence of an optimum number of features to use for jet classification. The recognition rates show no consistent dependence on the number of features, except for run 3 which improved as the number of features increased.

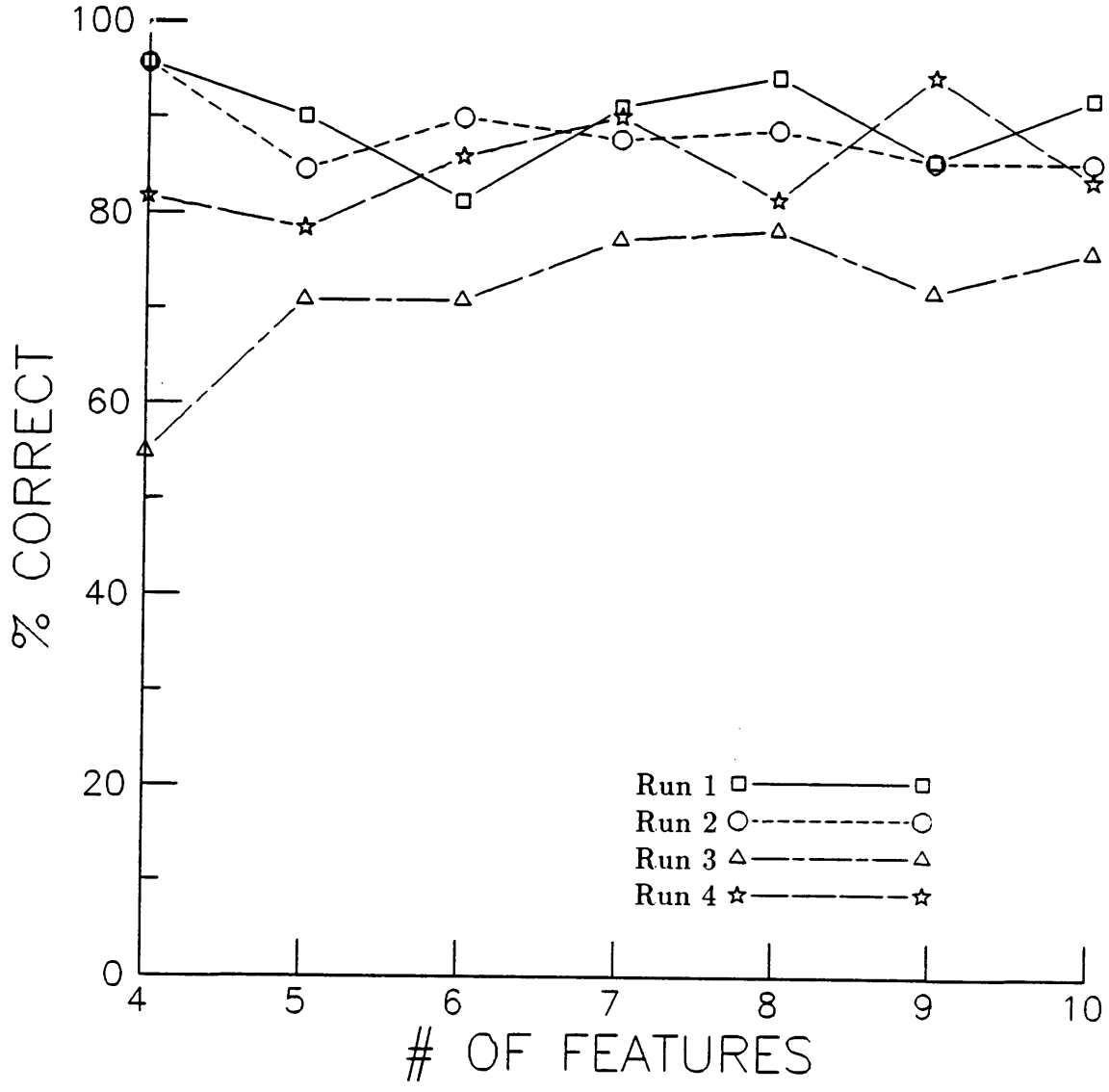

Figure 22. Jet Plane Recognition

Runs 3 and 4, both of which used single level structures, started off poorly, identifying only 55 percent and 81 percent of the jets, respectively. As the number of features increased, both improved in recognition, to a high of 95 percent for run 4 and 78 percent for run 3. The overall recognition rate of run 3 was hurt by its inability to correctly identify the jets.

5.4.3 Propeller Plane Recognition

The classification system could not identify propeller planes as well as the jets. The recognition rates for the classifiers went as low as 57 percent in run 2, but as high as 100 perent in run 3, shown in Fig. 23. The performance of the other runs and other number of features was not as high as in the jets. The propeller planes made up only 16 percent of the sources, but along with the 53 percent made up by the jets, fixed wing aircraft comprised nearly 70 percent of the test sources.

The propeller plane recordings contained planes recorded on takeoff and landing, but only the takeoff recordings were used for training. The recognition of the propeller planes suffered due to the difference between the training and testing sets, which will be discussed more fully in section 5.6.

The recognition rates of the different runs varied as the number of features changed, but not in any consistent fashion, with the exception of run 3. This run recognized more propeller planes as the number of features increased.

The recognition rates of the propeller planes varied over a wider range than those of the jets. Referring back the the description of the data base, the propeller planes recordings were of a large number of different types, but only a few recordings of each, so there was not an extensive training set as with the jets. This meant the cluster generated by these training sources had to contain a large amount of variation using only a few patterns to describe that variation. Figure 23 shows how this

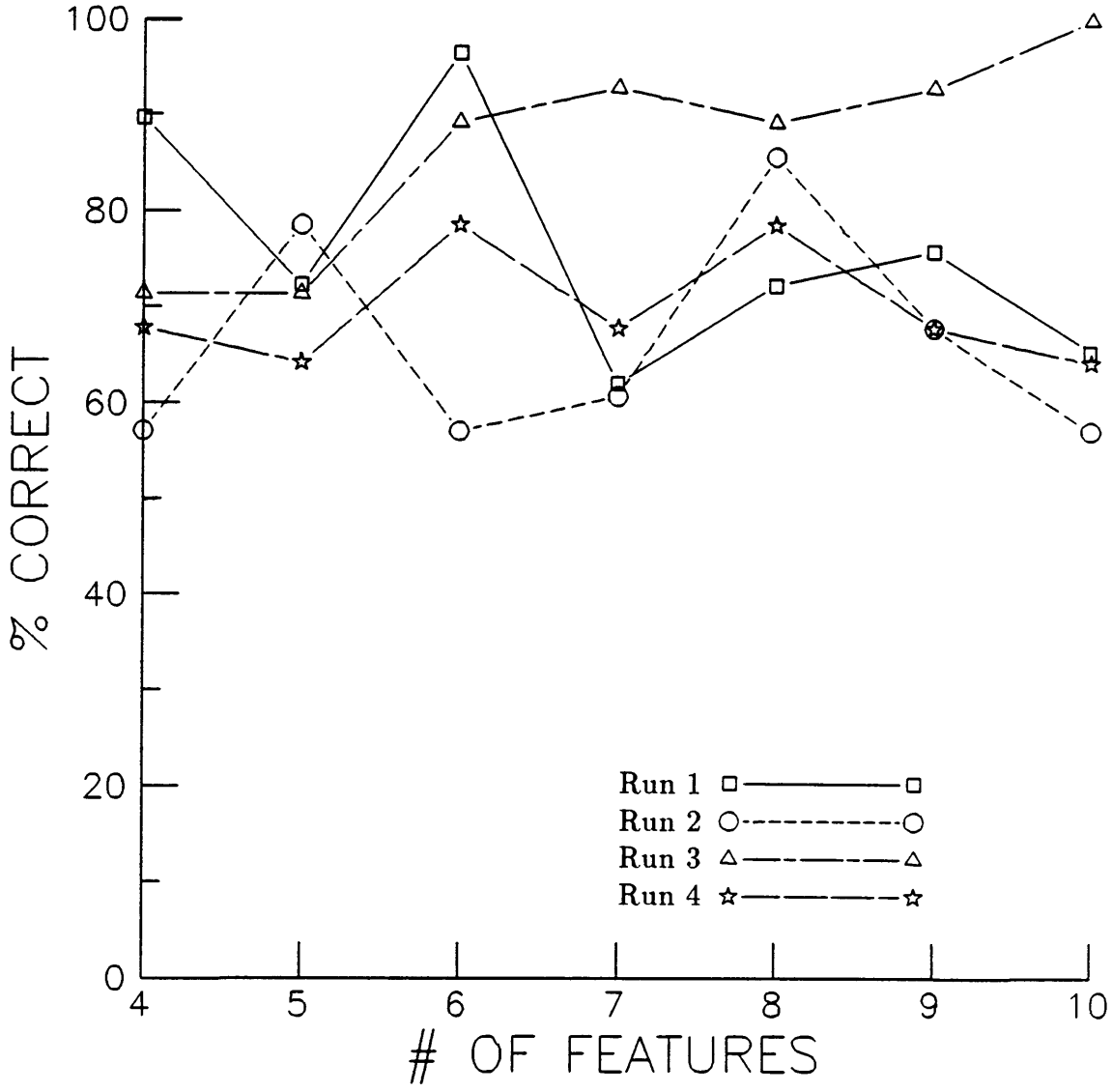

Figure 23. Propeller Plane Recognition

can produce varying results, depending on which features were selected and how well they described the propeller planes.

The plot in Fig. 23 shows there were no differences between the tree and single level structures. The performance of the single level structure, run 3, increased with the number of features while the performance of the other single level structure, run 4, decreased.

5.4.4 Wind Turbine Recognition

The wind turbine recordings were the most difficult for the classifiers to correctly identify on a consistent basis. Figure 24 illustrates how the recognition rates jumped around as the number of features changed. Wind Turbine recordings made up approximately 12 percent of the test sources.

A good example of the jumpiness of the recognition rate is run 3. The run began by identifying 80 percent of the recordings with 4 features, but immediately dropped to 5 percent when 5 features were used. Recognition dropped to 0 percent as 6 and 7 features were used. However, when 8 features were used, 100 percent of the recordings were correctly identified, but as before, the recognition rate changed drastically as the number of features changed to 9. The run ended with 100 percent correct identification. The other runs show a similar behavior as the number of features changed.

The rapid changes in recognition rate as the number of features changed indicated that performance was very sensitive to the type of features selected. Recall that for a given run, the same recording was identified by each classifier with 4 - 10 features. Figure 24 shows how performance on the same recording varied drastically as the number and type of features changed. These drastic changes indicate that correct identification was sensitive to the type of features used by a classifier.

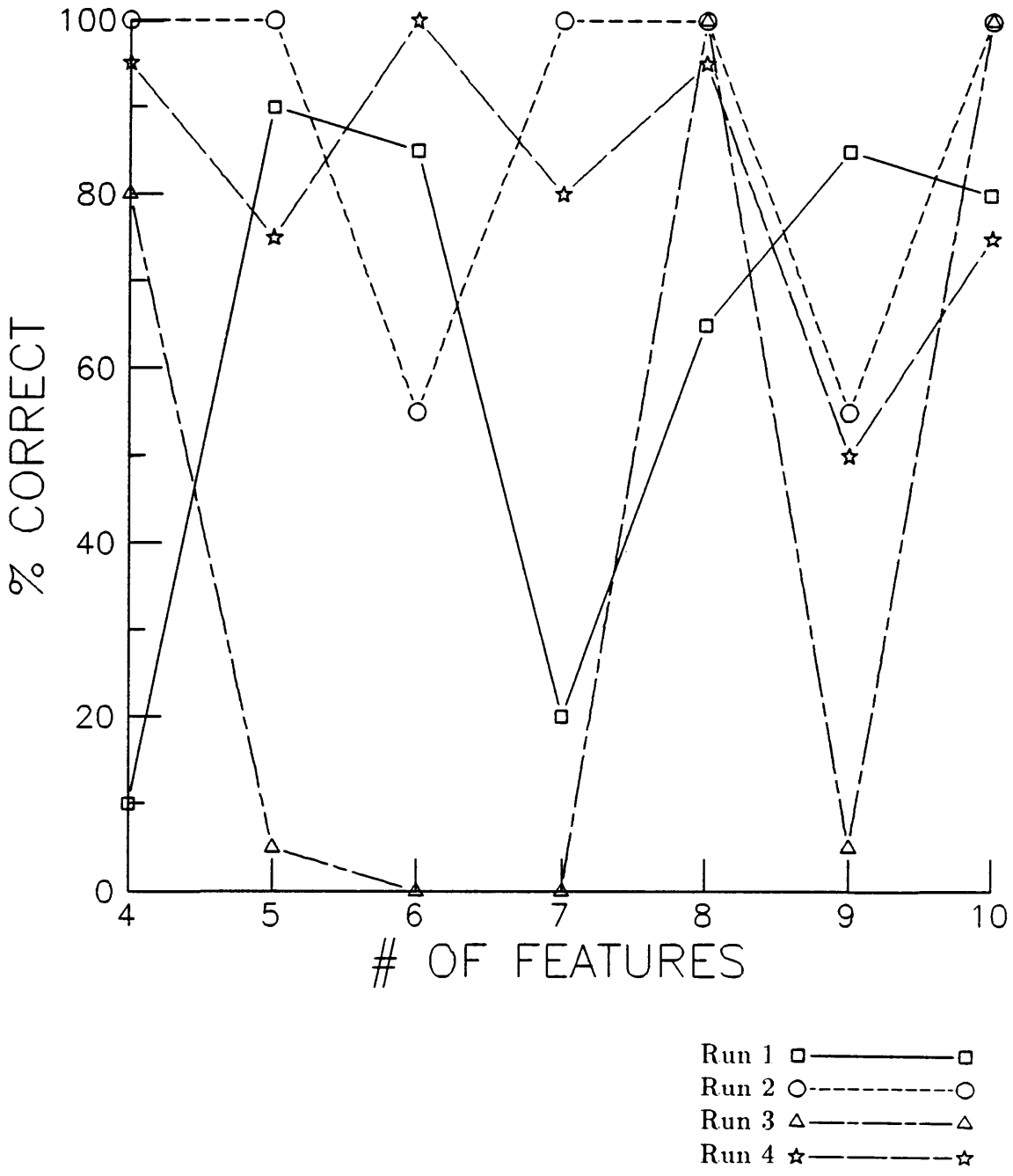

Figure 24. Wind Turbine Recognition

Classification performance should be sensitive to the features used for classification, but training patterns and the feature selection algorithm had been used to pick useful features. The recognition rates on the other sources suggest the feature selection algorithm was working, so the wind turbine must have had some unique characteristics making the results so sensitive.

The wind turbine was unique among the sources in that its signature was a very low level signal, corrupted by background noise. In contrast, the planes and trains had very loud recorded signatures that drowned out any background noise. This fact, coupled with the apparent success of the feature selection algorithm, suggests that the classifiers had difficulty identifying the noise corrupted signal of the wind turbine. The background noise confounded the identification process, with some features more sensitive to this effect than others.

Neither structure offered resistance to this noise effect, but the type of features may have offered some resistance. The recognition rates of runs 1 and 3 varied the most, and these two runs each used the pseudo features. The rates of runs 2 and 4 also varied with the number of features, but neither of their recognition rates dropped below 50 percent. The performance of the different features is compared in section 5.5, so discussion on this point is saved until then.

5.4.5 Train Recognition

The recognition rates on the train recordings were the highest of any source. Twelve classifiers achieved 100 percent recognition of the trains, while the lowest performer dropped to only 67 percent, as indicated in Fig. 25. The trains made up just over 10 percent of the test sources, so excellent performance on this class was not as strongly reflected in the overall rates as with the jet planes.

The recordings of the train had very little background noise, in contrast to those of the wind turbine. The noise of the train itself was constantly varying, since at any given time a different coal

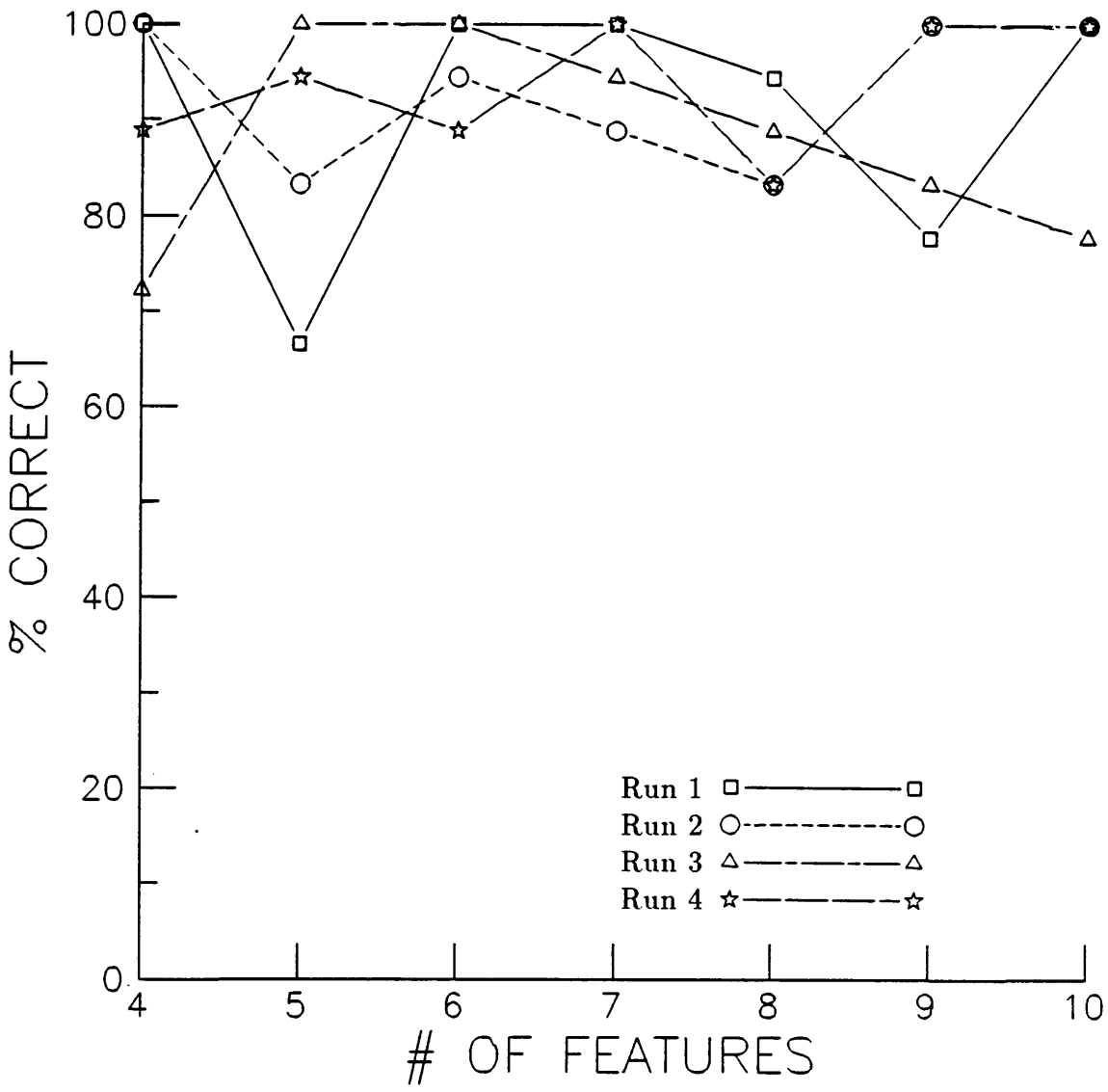

Figure 25. Train Recognition

car was rolling by with its own noise characteristics. The high recognition rates on the train support the claim made in the previous section that a signal corrupted by noise was difficult to identify. Despite the variations between the different cars, the presence of the strong signal with little background noise allowed the classifier to select a good set of descriptive features of train noise.

The tree structures, runs 1 and 2, correctly identified 100 percent of the train recording with only 4 features, but both were surpassed by the single level structures when the number of features increased to 5. The number of features seemed to have had little effect on recognition performance, as noted with the previous sources. Run 3 decreased in performance as the number of features increased, but runs 1, 2, and 4 recognized 100 percent of the recordings when 10 features were used.

5.4.6 Helicopter Recognition

The helicopter test recordings made up only 8 percent of the test sources. The recognition rates of the different runs are shown in Fig. 26.

As with the train, 12 classifiers correctly identified 100 percent of the helicopter recordings, but one run performed poorly on this class. Run 2, with 7 and 10 features correctly identified only 43 and 57 percent of the recordings, respectively. The helicopter recordings did not have as good a signal to noise ratio as the train and plane recordings, but they did not contain the level of backgound noise of the wind turbine recordings. The complexity of the feature selection and weight vectors prevents an accurate determination of the cause, but it is possible that the features for run 2 were more sensitive to the noise in the helicopter recordings.

The performance of the two structures varied with the number of features. Runs 1 and 2, using the tree structure, correctly identified 100 percent of the recordings with 8 and 9 features, while runs 3 and 4, using the single-level structure, correctly identified only 92 and 85 percent with 8 and 9

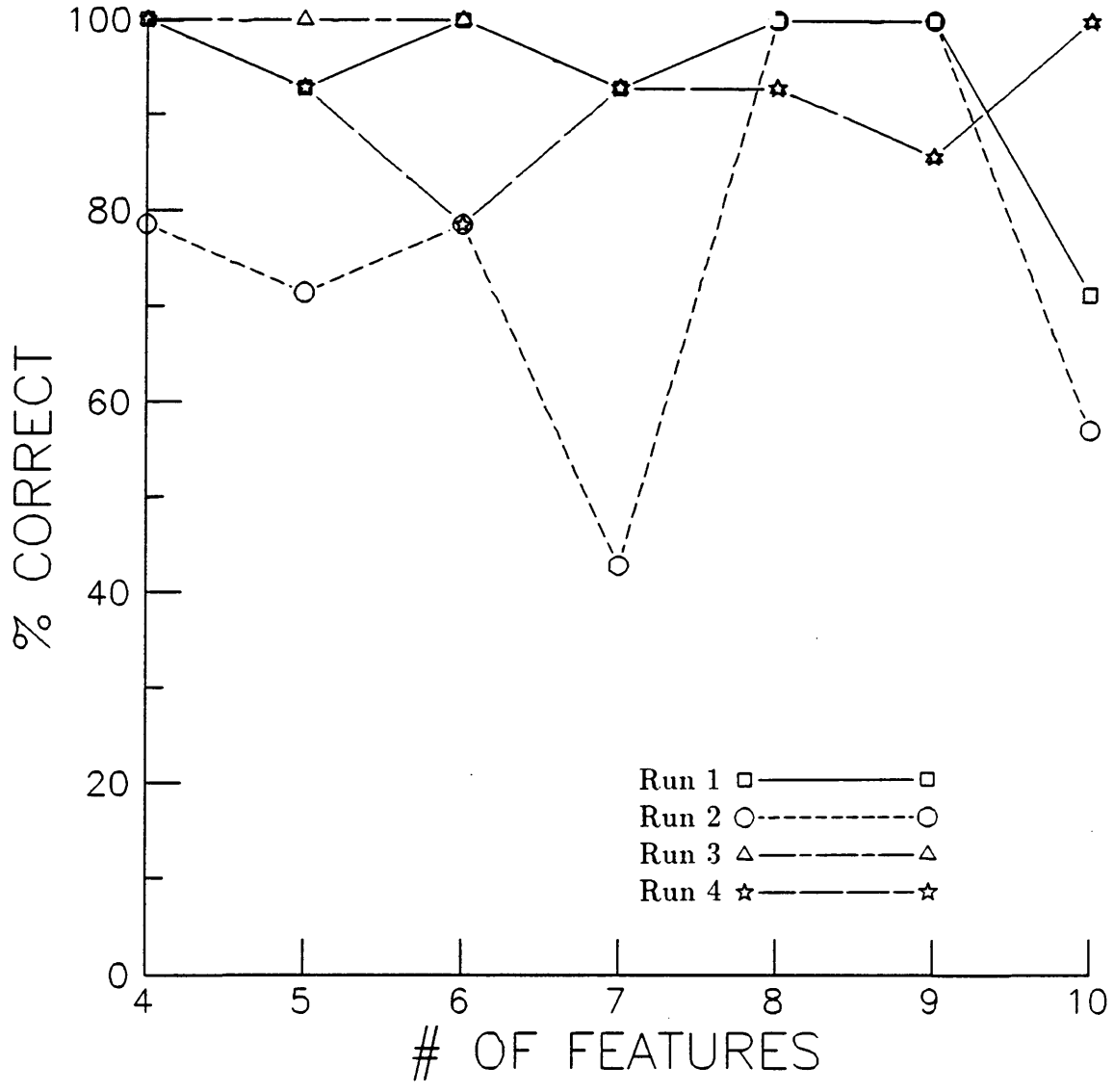

Figure 26. Helicopter Recognition

features, respectively. With ten features, however, runs 3 and 4 identified 100 percent of the test sources while runs 1 and 2 dropped below 75 percent correct identification.

5.5 Comparison of Tree and Single-Level Structures

Based on the overall recognition rates, the tree structure performed slightly better. The highest performer, run 2 with 8 features, used a tree structure, as opposed to the poorest performer, run 3 with 4 features, which used a single level structure.

In general, the results gave little indication that one structure was better than the other for identifying the test recordings. The recognition rates of the different runs varied much more with the number of features than with the type of structure. The tabulation of results in the appendix confirms the fact that for this particular application, either a single level structure or tree structure will give satisfactory recognition performance.

A better test of the structures would be to increase the number of classes to a more realistic level. Increasing the number of classes would place an increased burden on the single level classifier; as the number of classes grew, so would the decision functions and the number of features required for satisfactory performance. At some point, the required precision of the features would become impractical and classification performance would break down [5]. The tree classifiers make fewer classification errors on each level, so they are less sensitive to an increasing number of classes. All that is required of a tree structure is that a new branch or node in the tree be created.

5.6 Comparison of Feature Types

The two types of features, pseudo and regular, described in section 5.1, also seem to have had no effect on the performance of the classification system, with the exception of the wind turbine identification.

The wind turbine was unique in that all of its recordings were corrupted by background noise. The correct identification of the wind turbine's signature was difficult since some features were more sensitive to the background noise than others. Figure 24 shows the recognition performance of the runs on the wind turbine recordings. Runs 1 and 3 used the pseudo features, while 2 and 4 used the regular features. The recognition rates of runs 2 and 4 jumped around less than those of runs 1 and 3. While runs 1 and 3 varied from 0 percent recognition to 100 percent recognition, runs 2 and 4 varied between 50 and 100 percent recognition. The runs using the regular features appeared to be less sensitive to the broadband noise in the wind turbine recordings than the pseudo features.

The regular features worked better because they emphasized the low frequency energy of the wind turbine's signature. If the spectrum was corrupted by noise, that noise would be present throughout the frequency spectrum. The regular features would be less sensitive to the noise than the pseudo because the added energy from the noise would be dwarfed by the energy of the wind turbine signature at the low end of the spectrum.

The performance of runs 2 and 4 cannot be totally attributed to better performing regular features. Since the features selected and used in the classifiers varied from run to run, runs 2 and 4 could have performed better simply because they used better features than 1 and 3. The reason for claiming that the feature type affected the performance is that the overall performance was not as varied and chaotic for the regular features as the overall performance was for the pseudo features.

As with the two different structures, the benefits of one type of feature may become apparent only when more classes are included. The inclusion of more classes would require the features to be more precise since greater information would have to be conveyed by each feature. One of the methods of calculating features may then give an advantage, but for this application there appears to be none.

5.7 Robustness of Classifier

The robustness of the classifcation system is the ability of the classification system to correctly identify sources that are different from those in the training set. The ability to recognize sources with variation is an important property since a training set will rarely contain all the sources and conditions the system will encounter.

A robust classifier using the clustering approach will have to generate clusters in training that contain the operational variations of the sources. The word contain is used to indicate that a pattern vector generated by a source with some operational variation will have to fall into the bounds of the original class cluster in order to be identified correctly.

Recordings of propeller planes were used to test the ability of the classifier to identify sources very different from those used for training, in a measure of the robustness of the classification system. The ability of clusters generated by planes taking off to identify the same planes landing was investigated using recordings from the tape of propeller planes. The training sources were of takeoff, while the testing sources contained both takeoffs and landings. The noise generated by the planes would be different in each situation due to variations such as propeller pitch and engine speed. The conditions of the testing were identical; these sources were part of the 170 used for

testing and were not considered a separate set for the calculation of the recognition rates previously discussed.

Table 6 shows the recognition rates in percent, of the two different classes of propeller planes. The takeoff rate was calculated by summing the errors made with 4-10 features on those recordings similar to the training recordings. The landing rate was calculated by summing the errors made with 4-10 features on the same planes recorded landing.

The results in the table indicate that the classifiers could not recognize planes landing as well as planes taking off. In each run the percentage of planes correctly identified dropped for the planes landing. This indicates the clusters generated by the takeoff noise of propeller planes will not necessarily be useful for identifying the same planes landing.

The robustness of the cluster separation algorithm depends on the training data. The jets were identified with a high success rate since a wide variation was used for training. The propeller planes were identified with a lower success rate because a wide variety of sources was not used for training and the cluster was not useful for describing varying operating conditions.

The classifiers that used the pseudo features, runs 1 and 3, were more robust than the other classifiers. Because the pseudo features masked the true magnitude of an energy shift, they were not as affected by the shift that occurred with the planes. The regular features tended to magnify any energy shifts, so a shift in operating conditions affected the relative feature values.

5.8 Comments on the Performance in General

The results of testing illustrate the importance of the feature selection step in the performance of the classifier. As stated in [5], "real computational power often comes from a careful choice of the

Table 6. Recognition of Propeller Planes Landing vs Taking Off.

Recognition Rates (percent)		
Run #	Takeoff	Landing
1	81	66
2	91	37
3	90	80
4	94	41

attributes based on a good knowledge of the domain, rather than from the specific design of the separation algorithm."

The plot of the recognition rates of the wind turbine illustrates the effect different features have on performance. One set of features incorrectly identified all the recordings, while another set, using the same classifier and training set, correctly identified every recording. The problems of feature selection can best be met with a well designed training set for the problem at hand. The jets were recognized with a high success rate, despite the variations in types, because an extensive training set had been used to describe the variations in the class.

Ideally, the designer could test different structures and combinations of features and say that x number of features gives best recognition performance on average. The recognition rates of the classifiers in this study followed no pattern as to the performance as a function of number of features. This prevents the pre-specification of the optimal number of features, requiring extensive testing of each type of run on sources that would be expected to be encountered by the system to specify the best number of features for each classifier.

6.0 Conclusions

The object of this study was to determine the ability of a computer based pattern recognition system to identify aerospace acoustic sources using information from a single microphone. Such a system is envisioned as being used for the monitoring of air traffic in and around congested urban areas for the enforcement of noise guidelines.

The system was trained to recognize several different commercial jet planes, propeller planes, a helicopter, a train, and a wind turbine. The user provided a list of features that could have relevance to classification of the sources and during the training mode, the system automatically selected a good set of features for each classification task.

The classification ability of the system was tested by presenting 170 recordings of sources and tabulating the identifications made by the system. The errors and successes in identification resulted in recognition rates of the different classifiers. From these recognition rates, the following conclusions were made:

1. Linear decision functions trained with a perceptron algorithm were successful in correctly identifying the test sources. An overall success rate of 90 percent was achieved using a tree

arrangement of the decision functions and eight features calculated from the actual energy in the PSD to describe the sources.

2. The number of features had no effect on the recognition rate. The optimal number of features could not be determined from the results since performance varied markedly as the number of features changed.

3. The decision functions were successful at identifying members of a class that contained variation if the training sources also contained that variation. Recognition rates as high as 95 percent were achieved on the jet planes, which included 9 different commercial airliners and one military jet.

4. The performance of the tree and single level structures was equivalent on the sources used for testing.

5. The two types of features, pseudo and regular, offered no advantages over one another in terms of general recognition, but each improved recognition in certain situations.

    - The regular features, were less sensitive to background noise in identifying the wind turbine. The regular features described the actual distribution of energy in the spectrum so that when broadband background noise was present, the large amount of energy in the very low frequency dwarfed the small amount of energy from the noise, which changed the values of the features only slightly. In contrast, the pseudo features calculated a pseudo energy that did not express the large amount of energy in the low frequency range, so that broadband background noise changed the values of these features enough to impair recognition.

    - The pseudo features were less sensitive to variations between the training and testing sources. Recognition of propeller planes landing instead of taking off, as used in training, dropped from 81 to 66 percent and 90 to 80 percent in the two runs using the pseudo

features. Recognition of the same planes dropped from 91 to 37 percent and 94 to 41 percent in the two runs using regular features. The changes in the location of energy due to the different operating conditions was magnified by the regular features. Any peaks of energy dominated the bands around them so a change in the location of that energy produced a large change in the values of the features. The pseudo features were not so sensitive to concentrations of energy so a peak shift in the spectrum did not produce such a large change in the feature values.

6. The decision functions could not identify sources that differed substantially from those used in training as well as they identified sources similar to those used in training. The recognition rates were lower on propeller planes landing, as opposed to taking off as used in training.

7. Signals that were corrupted by background noise, such as the wind turbine recordings, were difficult to consistently identify correctly. Recognition rates varied from 0 percent correct to 100 percent correct when the features used for classification were changed.

7.0 Recommendations

A system designed to identify aerospace acoustic sources must be able to categorize a large variety of aircraft into a small number of classes, then further categorize the sources if necessary. The system must be very robust to large variations in the spectra of similar class members, suggesting the use of symbolic type features, rather than strictly numeric features. The system described in this study can be improved in terms of overall recognition rates with a few modifications.

1. The performance of the feature selection algorithm should be examined. The wide range of recognition rates on some of the sources indicates the feature selection routine does not always select an optimal set. The parameters that determine whether or not a feature is accepted in the final set should be examined, and changed to make the criterion more difficult. This includes the threshold for acceptance, and the number of passes through the data. Both should be increased, which will increase the time required for feature selection, but the improvement in results should justify the extra time.

2. The features themselves may not adequately convey recognition information or may be to correlated. The use of a large number of related features interferes with the feature selection because related features give similar classification results and are not needed. The frequency range should be divided into a smaller number of bandwidths instead of the large number used

for the features discussed in this study. The smaller number will be less precise about location of tones and harmonics, but they will be less correlated than many features describing the same frequency range, which should improve the feature selection. Features deriving information from a different source should also be investigated.

3. A more extensive training set should be used, covering a broader range of operating conditions, or conditions that have a high probability of being encountered by the system. The jets were correctly identified with a high success rate, and this class had the most extensive training set. A thorough set for each source will improve the robustness of the classifier.

4. The system architecture could be redesigned to enhance the generalization properties, such as a different tree structure, or the use of different classifiers at different levels of the structure.

5. The possibility of using a neural network, or several, in an acoustic source identification system should be investigated. The optimal configuration could be a tree structure of neural networks, each assigned to a particular piece of the identification task. Recent work suggests neural networks have generalization abilities suitable for the classification task.

6. Multiple microphones, arranged in an array configuration, could be used to provide additional information on the sources, such as source speed and heading, as well as providing additional sensitivity.

References

1. Moukas, P., Simson, J., and Norton-Wayne, L., "Automatic Identification of Noise Pollution Sources," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-12, No. 5, Sept/Oct 1982.

2. Hemdal, J.F., *An Investigation of Automatic Identification of Acoustic and Seismic Sources*, Willow Run Laboratories Institute of Science and Technology, University of Michigan, Ann Arbor, 1973.

3. Horvath, P., and Cook, F.J., "Establishing Signal Processing and Pattern Recognition Techniques for Inflight Discrimination Between Crack-Growth Acoustic Emissions and Other Acoustic Waveforms," *Review of Progress in Nondestructive Evaluation*, Vol. 1, Plenum Press, New York, 1982.

4. Chan, R.W.Y., and Hay, R.D., "A Case Study of Sensitivity of Some Pattern Classifiers Used in Sorting Acoustic Emission Signals," *Review of Progress in Quantitative Nondestructive Evaluation* , Vol. 1, Plenum Press, New York, 1982.

5. Chandrasekaran, B., and Goel, A., "From Numbers to Symbols to Knowledge Structures: Artificial Intelligence Perspectives on the Classification Task," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 18, No. 3, May/June 1988.

6. Tou, J.T., and Gonzalez, R.C., *Pattern Recognition Principles*, Addison-Wesley, Reading, 1974.

7. Bow, S.T., *Pattern Recognition - Applications to Large Data Set Problems*, Marcel Dekker, New York, 1984.

8. Lee, H.C., and Fu, K.S., "A Syntactic Pattern Recognition System With Learning Capability," *Information Systems*, Ed. Tou, J., Plenum Press, New York, 1972.

9. Moser, J.M., and Aunon, J.I., "Classification and Detection of Single Evoked Brain Potentials Using Time-Frequency Amplitude Features," *IEEE Transactions on Biomedical Engineering*, Vol. BME-33, No. 12, December 1986.

10. Peters, R.J., "Zero Order and Nonzero Order Decision Rules in Medical Diagnosis," *IBM Journal of Research and Development*, September 1977.

11. Bremermann, H.J., "What Mathematics Can and Cannot Do for Pattern Recognition," *Pattern Recognition in Biological and Technical Systems, Proceedings of the 4th Congress of the Deutsche Gesellschaft fur Kybernetik*, Eds. Grusser, O.J., Klinke, R., Springer-Verlag, Berlin, 1971.

12. Rosen, J.B., "Pattern Separation by Convex Programming," *Journal of Mathematical Analysis and Applications*, No. 10, 1965.

13. Forsyth, R., and Rada, R., *Machine Learning: Applications in Expert Systems and Information Retrieval*, Ellis Horwood Ltd., 1986.

14. McLelland, J.L., et al., *Parallel Distributed Processing*, Vol. 1, MIT Press, Cambridge, 1986.

15. Minsky, M.L., and Papert, S., *Perceptrons*, Expanded edition, MIT Press, Cambridge, 1988.

16. Klassen, G.S., *A General-Purpose Reduction-Intensive Feature Selector for Pattern Classification*, Master's Thesis, VPI&SU, 1986.

17. Thomas, D.W., "Vehicle Sounds and Recognition," *Pattern Recognition: Ideas in Practice*, Ed. Batchelor, B.G., Plenum Press, New York, 1978.

18. Sejnowski, T.J., and Rosenberg, C.R., "Parallel Networks that Learn to Pronounce English Text," *Complex Systems*, Vol. 1, Complex Systems Publications, 1987.

19. McCurdy, D.A., and Powell, C.A., *Annoyance Caused by Propeller Airplane Flyover Noise*, NASA TP-2356, August 1984.

20. Bendat, J.S., and Piersol, A.G., *Random Data: Analysis and Measurement Procedures*, 2nd ed., J. Wiley and Sons, New York, 1986.

21. *IMSL User's Manual*, Version 1.0, IMSL Inc., April 1987.

Appendix A. Training and Evaluation Sets

This section contains a listing of those recordings used for the training and evaluation sets of the classification system. The list below contains a number and short description of each run used. The numbers such as 261, 262, etc. correspond to numbers used in processing the recordings and are only used to keep track of the recordings. The column headings, 1T, 1E, etc., refer to the classifier the patterns were used to train. Runs 1 and 2 used the same sets of training and evaluation patterns since they shared similar structures, so the training sets are labeled as 1 only. Similarly, runs 3 and 4 had the same structures so they used the same training and evaluation sets, labeled 3 only. The notation in the tables is described below.

- 1T,1E - Training (T) and evaluation (E) patterns used for the top level classifier in the tree structure (See Fig. 11 in the text). This classifier identified a source as being either ground or aerospace.
- 1aT,1aE - Sets used on second level of tree structure, for identifying train and wind turbine sources.
- 1bT,1bE - Second level classification of helicopter versus fixed-wing aircraft.
- 1cT,1cE - Third level classification of propeller planes versus jet planes.
- 3T,3E - Training and evaluation patterns used for the single level structures; all five classes are included in these training and evaluation sets.

Each source is followed by a series of x's underneath the headings 1T,1aT, etc. The presence of an 'x' underneath a heading indicates the source was used in that particular training set. For

example, the second source, T-38 run 2, was used in the training sets of run 1 b and c. The source labels in this table do not correspond with the labels used for the testing sources, except for the planes. The helicopter, train, and wind turbine recordings were partitioned differently in training and testing so the labels do not correspond.

Run #	Description	1T	1E	1aT	1aE	1bT	1bE	1cT	1cE	3T	3E
261	T-38 run 1										
262	T-38 run 2					x		x			
263	T-38 run 3	x					x		x		x
264	T-38 run 4		x				x	x		x	
265	T-38 run 5					x					x
266	T-38 run 6		x				x	x			
267	T-38 run 7	x					x	x		x	
268	T-38 run 8					x			x		x
269a	Wind Turbine run 1a	x		x						x	x
269b	Wind Turbine run 1b		x		x						x
270a	Wind Turbine run 2a	x		x							x
270b	Wind Turbine run 2b		x		x					x	x
271a	Wind Turbine run 3a	x		x							x
271b	Wind Turbine run 3b		x		x					x	x
272a	Wind Turbine run 4aa	x		x						x	x
272b	Wind Turbine run 4ab		x		x					x	x
273a	Wind Turbine run 4ba	x		x						x	x
273b	Wind Turbine run 4bb		x		x						x
274a	Wind Turbine run 4ca	x		x						x	x
274b	Wind Turbine run 4cb		x		x						x
275	Helicopter run 1									x	x
276a	Helicopter run 2a	x				x	x				x
276b	Helicopter run 2b		x				x			x	x
277a	Helicopter run 3a		x			x	x			x	x
277b	Helicopter run 3b	x					x			x	x
278a	Helicopter run 4aa	x				x	x			x	x
278b	Helicopter run 4ab	x				x	x				x
279a	Helicopter run 4ba		x			x	x			x	x
279b	Helicopter run 4bb		x			x	x			x	x
280	Prop Plane flyover		x				x	x	x		x
281a	Train run 1a	x		x							x
281b	Train run 1b		x		x					x	x
282a	Train run 2a	x		x						x	x
282b	Train run 2b		x		x						x
282c	Train run 2c	x		x						x	x
283a	Train run 3a		x		x					x	x
283b	Train run 3b	x		x							x
283c	Train run 3c		x		x					x	x
284a	Train run 4a	x		x						x	x
284b	Train run 4b		x		x						x
284c	Train run 4c	x		x						x	x

Run #	Description	1T	1E	1aT	1aE	1bT	1bE	1cT	1cE	3T	3E
287	DC-8 TF run 1						x	x			
288	DC-8 TF run 2					x					x
289	DC-8 TF run 3		x				x		x	x	
290	DC-8 TF run 4	x				x		x			
291	DC-8 TF run 5						x		x		x
292	DC-8 TF run 6		x					x			
293	DC-8 TF run 7	x									x
294	747 run 1	x				x			x		
295	747 run 2	x					x	x			x
296	747 run 3		x				x	x			x
297	747 run 4		x			x			x	x	
298	747 run 5							x			
299	747 run 6						x		x		x
400	747 run 7										
401	L-1011 run 1	x						x			
402	L-1011 run 2		x				x		x		x
403	L-1011 run 3					x		x			x
404	L-1011 run 4					x		x		x	
405	L-1011 run 5		x				x		x		x
406	L-1011 run 6	x									
407	DC-10 run 1		x			x			x		x
408	DC-10 run 2	x					x			x	
410	DC-10 run 4		x					x			
411	DC-10 run 5										
412	DC-10 run 6	x					x	x			
413	DC-10 run 7					x			x		x
414	DC-10 run 8						x	x			x
415	727-100 run 1		x								
416	727-100 run 2	x							x		x
417	727-100 run 3	x						x			
418	727-100 run 4					x				x	
419	727-100 run 5						x				x
420	727-100 run 6						x		x		
421	727-100 run 7							x			x
422	727-100 run 8		x					x			
423	727-100 run 9						x			x	
424	727-100 run 10					x					
425	727-100 run 11						x				x
426	727-100 run 12								x		
427	727-100 run 13										
428	727-200 run 1						x				
429	727-200 run 2	x					x	x			x
430	727-200 run 3					x			x		
431	727-200 run 4		x					x		x	
432	727-200 run 5					x					x
433	727-200 run 6	x					x		x		
434	727-200 run 7		x					x			x
435	727-200 run 8						x		x		
436	727-200 run 9								x		

Run #	Description	1T	1E	1aT	1aE	1bT	1bE	1cT	1cE	3T	3E
437	737 run 1						x		x		x
438	737 run 2	x				x		x		x	
439	737 run 3		x			x			x		x
440	737 run 4	x					x		x		
441	737 run 5		x					x			
442	737 run 6							x			x
443	DC-9 run 1	x					x		x		x
444	DC-9 run 2					x		x			x
445	DC-9 run 3	x					x	x		x	
446	DC-9 run 4						x		x		
447	DC-9 run 5		x			x			x		
448	DC-9 run 6		x					x			x
449	DC-9 run 7								x		
450	DC-8 TJ run 1		x				x	x		x	
451	DC-8 TJ run 2	x				x			x		x
452	DC-8 TJ run 3	x				x		x			x
453	DC-8 TJ run 4		x				x		x		x
454	DC-8 TJ run 5										
455	DC-8 TJ run 6							x			
456	King Air		x			x		x	x		x
457	Nord 262	x					x	x	x	x	x
458	P-3		x			x	x	x	x	x	
459	EMB-110	x					x	x	x		x
460	Dash 7	x				x		x	x		x
461	Viscount	x				x	x	x	x	x	x
462	Shorts 330										
463	YS-11		x				x	x	x		x
464	C-130		x			x	x		x	x	x
465	Gulfstream		x				x		x		
466	Sw Metro	x					x		x	x	x
467	MU-2	x	x				x				
468	Cessna 152	x				x	x	x	x	x	x
469	Cessna 172		x			x	x	x	x	x	x
470	Supercub	x				x	x	x	x		x
471	Bonanza V	x				x	x		x	x	x
472	G. Tiger		x				x	x	x		x

Appendix B. Features Used for Identification

B.1 Calculated Features

This section containes a list of the features calculated from the PSD and autocorrelation. The numbers of the features are used in appendix B.2 to describe the features selected for each classifier. This list has a cyclical nature that illustrates how many features were calculated by changing only a few specifications. For example, the list begins with a set of features calculated in the 0-2500 Hz band. Features are calculated from this band until feature number 30, at which point the band changes to 0-500 Hz, and similar features are calculated from the new band. This continues with different bands, until the last two features which are calculated from the autocorrelation.

Feature #	Description
1	Number of peaks exceeding preset threshold in the PSD
2	Percent of power from 0-2500 Hz in 0-125 Hz band
3	Percent of power from 0-2500 Hz in 125-250 Hz band
4	Percent of power from 0-2500 Hz in 250-375 Hz band
5	Percent of power from 0-2500 Hz in 375-500 Hz band
6	Percent of power from 0-2500 Hz in 500-625 Hz band
7	Percent of power from 0-2500 Hz in 625-750 Hz band
8	Percent of power from 0-2500 Hz in 750-875 Hz band
9	Percent of power from 0-2500 Hz in 875-1000 Hz band
10	Percent of power from 0-2500 Hz in 1000-1125 Hz band
11	Percent of power from 0-2500 Hz in 1125-1250 Hz band
12	Percent of power from 0-2500 Hz in 1250-1375 Hz band
13	Percent of power from 0-2500 Hz in 1375-1500 Hz band
14	Percent of power from 0-2500 Hz in 1500-1625 Hz band
15	Percent of power from 0-2500 Hz in 1625-1750 Hz band
16	Percent of power from 0-2500 Hz in 1750-1875 Hz band
17	Percent of power from 0-2500 Hz in 1875-2000 Hz band
18	Percent of power from 0-2500 Hz in 2000-2125 Hz band
19	Percent of power from 0-2500 Hz in 2125-2250 Hz band
20	Percent of power from 0-2500 Hz in 2250-2375 Hz band
21	Percent of power from 0-2500 Hz in 2375-2500 Hz band
22	Ratio of percent power in 500-625 Hz band to 875-1000 Hz band
23	Ratio of percent power in 500-625 Hz band to 1875-2000 Hz band
24	Ratio of percent power in 875-1000 Hz band to 1875-2000 Hz band
25	Freq. at which 50 of accumulated power in 0-2500 Hz band was observed
26	Freq. at which 60 of accumulated power in 0-2500 Hz band was observed
27	Freq. at which 70 of accumulated power in 0-2500 Hz band was observed
28	Freq. at which 80 of accumulated power in 0-2500 Hz band was observed
29	Freq. at which 90 of accumulated power in 0-2500 Hz band was observed
30	Percent of power from 0-500 Hz in 0-50 Hz band
31	Percent of power from 0-500 Hz in 50-100 Hz band
32	Percent of power from 0-500 Hz in 100-150 Hz band
33	Percent of power from 0-500 Hz in 150-200 Hz band
34	Percent of power from 0-500 Hz in 200-250 Hz band
35	Percent of power from 0-500 Hz in 250-300 Hz band
36	Percent of power from 0-500 Hz in 300-350 Hz band
37	Percent of power from 0-500 Hz in 350-400 Hz band
38	Percent of power from 0-500 Hz in 400-450 Hz band
39	Percent of power from 0-500 Hz in 450-500 Hz band
40	Ratio of percent power in 50-100 Hz band to 450-500 Hz band
41	Freq. at which 50 of accumulated power in 0-500 Hz band was observed
42	Freq. at which 60 of accumulated power in 0-500 Hz band was observed
43	Freq. at which 70 of accumulated power in 0-500 Hz band was observed
44	Freq. at which 80 of accumulated power in 0-500 Hz band was observed
45	Freq. at which 90 of accumulated power in 0-500 Hz band was observed

Feature #	Description
46	Percent of power from 500-1000 Hz in 500-525 Hz band
47	Percent of power from 500-1000 Hz in 525-550 Hz band
48	Percent of power from 500-1000 Hz in 550-575 Hz band
49	Percent of power from 500-1000 Hz in 575-600 Hz band
50	Percent of power from 500-1000 Hz in 600-625 Hz band
51	Percent of power from 500-1000 Hz in 625-650 Hz band
52	Percent of power from 500-1000 Hz in 650-675 Hz band
53	Percent of power from 500-1000 Hz in 675-700 Hz band
54	Percent of power from 500-1000 Hz in 700-725 Hz band
55	Percent of power from 500-1000 Hz in 725-750 Hz band
56	Percent of power from 500-1000 Hz in 750-775 Hz band
57	Percent of power from 500-1000 Hz in 775-800 Hz band
58	Percent of power from 500-1000 Hz in 800-825 Hz band
59	Percent of power from 500-1000 Hz in 825-850 Hz band
60	Percent of power from 500-1000 Hz in 850-875 Hz band
61	Percent of power from 500-1000 Hz in 875-900 Hz band
62	Percent of power from 500-1000 Hz in 900-925 Hz band
63	Percent of power from 500-1000 Hz in 925-950 Hz band
64	Percent of power from 500-1000 Hz in 950-975 Hz band
65	Percent of power from 500-1000 Hz in 975-1000 Hz band
66	Ratio of percent power in 500-525 Hz band to 725-750 Hz band
67	Freq. at which 50 of accum. power in 500-1000 Hz band was observed
68	Freq. at which 60 of accum. power in 500-1000 Hz band was observed
69	Freq. at which 70 of accum. power in 500-1000 Hz band was observed
70	Freq. at which 80 of accum. power in 500-1000 Hz band was observed
71	Freq. at which 90 of accum. power in 500-1000 Hz band was observed
72	Percent of power from 0-1000 Hz in 0-100 Hz band
73	Percent of power from 0-1000 Hz in 100-200 Hz band
74	Percent of power from 0-1000 Hz in 200-300 Hz band
75	Percent of power from 0-1000 Hz in 300-400 Hz band
76	Percent of power from 0-1000 Hz in 400-500 Hz band
77	Percent of power from 0-1000 Hz in 500-600 Hz band
78	Percent of power from 0-1000 Hz in 600-700 Hz band
79	Percent of power from 0-1000 Hz in 700-800 Hz band
80	Percent of power from 0-1000 Hz in 800-900 Hz band
81	Percent of power from 0-1000 Hz in 900-1000 Hz band
82	Ratio of percent power in 100-200 Hz band to 900-1000 Hz band
83	Ratio of percent power in 400-500 Hz band to 900-1000 Hz band
84	Freq. at which 50 of accum. power in 0-1000 Hz band was observed
85	Freq. at which 60 of accum. power in 0-1000 Hz band was observed
86	Freq. at which 70 of accum. power in 0-1000 Hz band was observed
87	Freq. at which 80 of accum. power in 0-1000 Hz band was observed
88	Freq. at which 90 of accum. power in 0-1000 Hz band was observed

Feature #	Description
89	Percent of power from 1000-2500 Hz in 1000-1150 Hz band
90	Percent of power from 1000-2500 Hz in 1150-1300 Hz band
91	Percent of power from 1000-2500 Hz in 1300-1450 Hz band
92	Percent of power from 1000-2500 Hz in 1450-1650 Hz band
93	Percent of power from 1000-2500 Hz in 1600-1750 Hz band
94	Percent of power from 1000-2500 Hz in 1750-1900 Hz band
95	Percent of power from 1000-2500 Hz in 1900-2050 Hz band
96	Percent of power from 1000-2500 Hz in 2050-2200 Hz band
97	Percent of power from 1000-2500 Hz in 2200-2350 Hz band
98	Percent of power from 1000-2500 Hz in 2350-2500 Hz band
99	Ratio of percent power in 1000-1150 Hz band to 2350-2500 Hz band
100	Ratio of percent power in 1150-1300 Hz band to 2200-2350 Hz band
101	Ratio of percent power in 1600-1750 Hz band to 2350-2500 Hz band
102	Freq. at which 50 of accum. power in 1000-2500 Hz band was observed
103	Freq. at which 60 of accum. power in 1000-2500 Hz band was observed
104	Freq. at which 70 of accum. power in 1000-2500 Hz band was observed
105	Freq. at which 80 of accum. power in 1000-2500 Hz band was observed
106	Freq. at which 90 of accum. power in 1000-2500 Hz band was observed
107	Average of autocorrelation across 768 msec
108	Standard deviation of autocorrelation across 768 msec

B.2 Selected Features by Classifier and Number of Features

The following tables give the feature numbers for each classifier as selected by the feature selection algorithm. The name of the particular classifier is given at the top of each table, and the numbers of the features are given below that. Each classifier was trained with 4-10 features, so the feature numbers for each set are given. For example, The ground vs. aerospace classifier, with 4 features, selected feature numbers 1, 72, 73,and 108. Looking back at appendix B.1, the features picked were the number of peaks exceeding a preset threshold in the PSD, the percent of power from 0-1000 Hz band in 0-100 Hz band, the percent of power from 0-1000 Hz band in 100-200 Hz band, and the standard deviation of the autocorrelation.

Ground vs Aerospace Classifier, Run 1										
# of Features	Selected Feature Numbers (from appendix B.1)									
4	1	72	73	108						
5	1	26	71	72	69					
6	1	30	92	99	100	107				
7	1	30	72	74	95	97	100			
8	1	2	17	21	30	55	81	97		
9	1	32	53	64	72	89	91	102	105	
10	1	7	23	30	34	62	81	93	95	100
Train vs Wind Turbine Classifier, Run 1										
# of Features	Selected Features									
4	17	27	83	93						
5	5	15	33	34	37					
6	5	41	48	90	93	108				
7	3	4	5	46	47	48	103			
8	4	5	16	38	46	55	67	79		
9	3	4	5	6	7	23	62	69	70	
10	4	5	6	7	8	22	24	32	69	107
Helicopter vs Plane Classifier, Run 1										
# of Features	Selected Features									
4	91	94	107	108						
5	30	46	70	105	107					
6	30	33	37	72	80	97				
7	30	46	54	68	81	84	107			
8	14	21	30	43	44	61	98	108		
9	9	30	32	35	47	54	72	82	90	
10	30	32	49	74	75	93	100	101	102	103
Propeller vs Jet Plane Classifier, Run 1										
# of Features	Selected Features									
4	1	27	41	42						
5	28	82	104	106	107					
6	29	36	37	48	95	108				
7	6	40	46	47	48	49	50			
8	47	56	64	77	78	90	96	107		
9	2	40	44	68	71	75	88	107	108	
10	5	6	40	41	48	66	79	83	93	107

Ground vs Aerospace Classifier, Run 2										
# of Features	Selected Features									
4	85	98	105	108						
5	44	97	98	100	107					
6	44	73	94	106	107	108				
7	57	81	85	86	100	102	103			
8	41	53	56	78	85	94	98	107		
9	33	40	77	84	94	101	102	106	108	
10	10	11	28	32	60	64	84	93	106	108
Train vs Wind Turbine Classifier, Run 2										
# of Features	Selected Features									
4	15	24	30	108						
5	46	47	73	84	108					
6	7	8	45	84	101	103				
7	42	46	51	52	66	85	93			
8	8	9	40	49	58	94	96	98		
9	7	10	63	64	68	81	82	93	106	
10	1	39	66	74	79	80	81	96	105	108
Helicopter vs Plane Classifier, Run 2										
# of Features	Selected Features									
4	30	88	91	103						
5	70	91	93	107	108					
6	42	43	78	85	93	106				
7	42	56	63	74	88	90	92			
8	20	54	55	57	69	71	87	105		
9	26	30	34	41	77	80	81	83	107	
10	6	9	10	16	23	24	30	66	92	93
Propeller vs Jet Planes Classifier, Run 2										
# of Features	Selected Features									
4	30	78	92	107						
5	55	60	78	93	107					
6	30	65	75	98	103	104				
7	30	60	81	87	96	98	107			
8	30	37	57	80	82	83	87	89		
9	25	30	51	62	70	75	76	104	108	
10	30	72	73	76	79	80	88	96	101	107

Single Level Classifier, Run 3										
# of Features	Selected Features									
4	21	30	48	65						
5	5	18	30	73	95					
6	30	32	35	36	61	89				
7	1	13	30	64	79	90	97			
8	20	21	30	36	63	73	93	107		
9	18	19	30	33	34	48	58	75	92	
10	5	7	18	30	34	35	49	81	91	107
Single Level Classifier, Run 4										
# of Features	Selected Features									
4	28	30	44	95						
5	24	28	42	59	107					
6	27	30	84	85	91	95				
7	27	72	84	95	99	104	107			
8	24	27	37	40	65	94	99	107		
9	26	28	30	40	81	91	99	100	107	
10	4	34	41	64	79	87	90	91	94	107

Appendix C. Results

C.1 Classification System Output

This section contains the outputs of the classifiers for the testing phase. The output is given for each test recording presented to the classifier, for runs 1-4 each with 4-10 features. The table shows the output for each source along with a brief description of the source. The actual class of the source is entered under the column labeled *Class*. A classifier's output for each number of features is shown in the next 7 columns, but only if the classifier incorrectly identified the source. So a blank entry signifies the classifier responded with the same class as the one in the column *Class*.

The classes are labeled as follows:

PR Propeller plane

J Jet plane

WT Wind turbine

HE Helicopter

TR Train

The runs were:

Run 1 Treed structure, non-dB features

Run 2 Treed structure, dB features

Run 3 Single level structure, non-dB features

Run 4 Single level structure, dB features

C.1.1 Test Results - Run 1

			Number of Features						
Source		Class	4	5	6	7	8	9	10
T-38	run 1	J							
T-38	run 2	J			PR				
T-38	run 3	J							
T-38	run 4	J							
T-38	run 5	J							
T-38	run 6	J							
T-38	run 7	J							
T-38	run 8	J							
Wind Turbine	run 1	WT		HE		TR		HE	J
Wind Turbine	run 2	WT	TR						
Wind Turbine	run 3	WT	TR						
Wind Turbine	run 4	WT	TR						
Wind Turbine	run 5	WT	TR			TR			
Wind Turbine	run 6	WT	TR						
Wind Turbine	run 7	WT	TR			TR	TR	TR	TR
Wind Turbine	run 8	WT	TR		HE	TR	TR		
Wind Turbine	run 9	WT	TR		HE	TR	TR		TR
Wind Turbine	run 10	WT			HE	TR			
Wind Turbine	run 11	WT	TR			TR	TR		
Wind Turbine	run 12	WT	TR			TR	TR		
Wind Turbine	run 13	WT	TR	TR		TR	TR	TR	TR
Wind Turbine	run 14	WT	TR			TR	TR		
Wind Turbine	run 15	WT	TR			TR			
Wind Turbine	run 16	WT	TR			TR			
Wind Turbine	run 17	WT	TR			TR			
Wind Turbine	run 18	WT	TR			TR			
Wind Turbine	run 19	WT	TR			TR			
Wind Turbine	run 20	WT	TR			TR			
Helicopter	run 1	HE							
Helicopter	run 2	HE							
Helicopter	run 3	HE							
Helicopter	run 4	HE							
Helicopter	run 5	HE							
Helicopter	run 6	HE							
Helicopter	run 7	HE		WT		TR			WT
Helicopter	run 8	HE							J
Helicopter	run 9	HE							

			Number of Features						
Source		Class	4	5	6	7	8	9	10
Helicopter	run 10	HE							J
Helicopter	run 11	HE							
Helicopter	run 12	HE							J
Helicopter	run 13	HE							
Helicopter	run 14	HE							
Train	run 1	TR							
Train	run 2	TR							
Train	run 3	TR							
Train	run 4	TR							
Train	run 5	TR							
Train	run 6	TR		WT					
Train	run 7	TR		WT				WT	
Train	run 8	TR							
Train	run 9	TR		WT				WT	
Train	run 10	TR							
Train	run 11	TR							
Train	run 12	TR							
Train	run 13	TR							
Train	run 14	TR							
Train	run 15	TR		WT					
Train	run 16	TR						WT	
Train	run 17	TR		WT					
Train	run 18	TR		WT			WT	WT	
NASA Test Plane		PR		J		J	J	J	J
King Air		PR				J			J
Nord 262		PR	J			J			
P-3		PR							
EMB-110		PR				J			J
Dash 7		PR							
B-727		J							PR
DC-9		J			PR		PR		
Viscount		PR							
Shorts 330		PR				J		J	J
YS-11		PR							
C-130		PR							
Swearingen Metro		PR		J	J	J	J		
B-707		J			PR	PR	PR		
DC-10		J	PR						
A-300		J						PR	PR
T-28		PR		J			J	J	J
King Air		PR	J			J	J	J	HE
Nord 262		PR		J		J	J	J	J
P-3		PR							
EMB-110		PR							
Dash 7		PR				J			HE
B-727		J			PR	PR		PR	PR

Source		Class	Number of Features						
			4	5	6	7	8	9	10
DC-9		J			PR	PR			
Viscount		PR		J				J	
Viscount		PR		J			J		
Shorts 330		PR	J	J		TR	J	J	J
YS-11		PR							HE
C-130		PR					J		
Swearingen Metro		PR		J		J			
B-707		J	PR	PR	PR	PR	PR	PR	HE
DC-10		J			PR	PR		PR	PR
A-300		J		PR	PR	PR		PR	PR
MU-2		PR							
Cessna 152		PR							
Cessna 172		PR							
Supercub		PR							
Bonanza IV		PR							
G. Tiger		PR							
727-100	run 1	J							
727-100	run 2	J							
727-100	run 3	J							
727-100	run 4	J							
727-100	run 5	J							
727-100	run 6	J							
727-100	run 7	J							
727-100	run 8	J							
727-100	run 9	J							
727-100	run 10	J							
727-100	run 11	J							
727-100	run 12	J							
727-100	run 13	J			PR				
727-200	run 1	J							
727-200	run 2	J							
727-200	run 3	J							
727-200	run 4	J							
727-200	run 5	J							
727-200	run 6	J							
727-200	run 7	J							
727-200	run 8	J						PR	
727-200	run 9	J							
737	run 1	J							
737	run 2	J	PR						
737	run 3	J							
737	run 4	J							
737	run 5	J							
737	run 6	J							

			Number of Features						
Source		Class	4	5	6	7	8	9	10
DC-9	run 1	J							
DC-9	run 2	J							
DC-9	run 3	J							
DC-9	run 4	J							
DC-9	run 5	J							
DC-9	run 6	J							
DC-9	run 7	J							
DC-8 Turbojet	run 1	J							
DC-8 Turbojet	run 2	J						PR	PR
DC-8 Turbojet	run 3	J							
DC-8 Turbojet	run 4	J							
DC-8 Turbojet	run 5	J							
DC-8 Turbojet	run 6	J							
DC-8 Turbofan	run 1	J							
DC-8 Turbofan	run 2	J							
DC-8 Turbofan	run 3	J							
DC-8 Turbofan	run 4	J							
DC-8 Turbofan	run 5	J							
DC-8 Turbofan	run 6	J							
DC-8 Turbofan	run 7	J		PR			PR	PR	
747	run 1	J							
747	run 2	J							
747	run 3	J							
747	run 4	J							
747	run 5	J							
747	run 6	J							
747	run 7	J							
L-1011	run 1	J							
L-1011	run 2	J							
L-1011	run 3	J							
L-1011	run 4	J							
L-1011	run 5	J		PR	PR			PR	
L-1011	run 6	J							
DC-10	run 1	J							
DC-10	run 2	J			PR			HE	
DC-10	run 3	J			PR				
DC-10	run 4	J			PR				
DC-10	run 5	J							
DC-10	run 6	J							
DC-10	run 7	J							
DC-10	run 8	J							
DC-10	run 9	J		PR					
DC-10	run 10	J		PR	PR				
DC-10	run 11	J	PR	PR	PR	PR	PR	HE	
DC-10	run 12	J		PR	PR			PR	
DC-10	run 13	J		PR	PR	PR		HE	

C.1.2 Test Results - Run 2

			Number of Features						
Source		Class	4	5	6	7	8	9	10
T-38	run 1	J							
T-38	run 2	J							
T-38	run 3	J							
T-38	run 4	J							
T-38	run 5	J							
T-38	run 6	J							
T-38	run 7	J							
T-38	run 8	J							
Wind Turbine	run 1	WT							
Wind Turbine	run 2	WT							
Wind Turbine	run 3	WT							
Wind Turbine	run 4	WT							
Wind Turbine	run 5	WT			TR			TR	
Wind Turbine	run 6	WT						TR	
Wind Turbine	run 7	WT			TR			TR	
Wind Turbine	run 8	WT			TR			TR	
Wind Turbine	run 9	WT			TR			TR	
Wind Turbine	run 10	WT			TR			TR	
Wind Turbine	run 11	WT			TR				
Wind Turbine	run 12	WT			TR			TR	
Wind Turbine	run 13	WT			TR			TR	
Wind Turbine	run 14	WT			TR			TR	
Wind Turbine	run 15	WT							
Wind Turbine	run 16	WT							
Wind Turbine	run 17	WT							
Wind Turbine	run 18	WT							
Wind Turbine	run 19	WT							
Wind Turbine	run 20	WT							
Helicopter	run 1	HE	TR			TR			
Helicopter	run 2	HE		TR	WT				TR
Helicopter	run 3	HE		J		J			J
Helicopter	run 4	HE	WT	TR	WT				TR
Helicopter	run 5	HE				J			
Helicopter	run 6	HE		TR	WT				
Helicopter	run 7	HE				J			J
Helicopter	run 8	HE				J			J
Helicopter	run 9	HE				J			

Number of Features									
Source		Class	4	5	6	7	8	9	10
Helicopter	run 10	HE							
Helicopter	run 11	HE	WT			TR			
Helicopter	run 12	HE				J			
Helicopter	run 13	HE							
Helicopter	run 14	HE				J			J
Train	run 1	TR							
Train	run 2	TR		WT					
Train	run 3	TR							
Train	run 4	TR							
Train	run 5	TR				WT			
Train	run 6	TR			WT				
Train	run 7	TR							
Train	run 8	TR					WT		
Train	run 9	TR		WT			WT		
Train	run 10	TR				WT			
Train	run 11	TR							
Train	run 12	TR							
Train	run 13	TR							
Train	run 14	TR							
Train	run 15	TR							
Train	run 16	TR		WT					
Train	run 17	TR							
Train	run 18	TR					WT		
King Air		PR							
Nord 262		PR							
P-3		PR				TR			
EMB-110		PR							
Dash 7		PR							
B-727		J							
DC-9		J				TR			
Viscount		PR			J				
Shorts 330		PR							
YS-11		PR				TR	TR		
C-130		PR							
Swearingen Metro		PR			J				HE
B-707		J							
DC-10		J							
A-300		J						PR	
T-28		PR	J	J	J	J	J	J	J
King Air		PR	J	J	J	J		J	HE
Nord 262		PR	TR	J	J	J		TR	J
P-3		PR	J		J	J		J	J
EMB-110		PR	J						J
Dash 7		PR	J		J			J	J
B-727		J	PR	PR	PR		PR	PR	

Number of Features									
Source		Class	4	5	6	7	8	9	10
DC-9		J		PR	PR		PR	PR	
Viscount		PR	J	J		J		J	J
Viscount		PR	J	J	J	J	J	J	J
Shorts 330		PR	J	J	J	J	J	J	J
YS-11		PR			J				J
C-130		PR	J		J				
Swearingen Metro		PR	J		J	J		J	J
B-707		J	PR	PR	PR	PR	PR	PR	PR
DC-10		J		PR	PR		PR	PR	
A-300		J		PR			PR	PR	
MU-2		PR	J						
Cessna 152		PR							
Cessna 172		PR							
Supercub		PR							
Bonanza IV		PR							
G. Tiger		PR							
727-100	run 1	J							
727-100	run 2	J							
727-100	run 3	J							
727-100	run 4	J							
727-100	run 5	J					PR		
727-100	run 6	J							
727-100	run 7	J							
727-100	run 8	J							
727-100	run 9	J							
727-100	run 10	J							
727-100	run 11	J							
727-100	run 12	J							
727-100	run 13	J							
727-200	run 1	J							
727-200	run 2	J						PR	
727-200	run 3	J							
727-200	run 4	J							
727-200	run 5	J							
727-200	run 6	J							
727-200	run 7	J							
727-200	run 8	J							
727-200	run 9	J		PR				PR	PR
727-200	run 10	J							
737	run 1	J	PR	PR			PR		
737	run 2	J					PR		
737	run 3	J							
737	run 4	J							
737	run 5	J							
737	run 6	J							

Number of Features									
Source		Class	4	5	6	7	8	9	10
DC-9	run 1	J							
DC-9	run 2	J							
DC-9	run 3	J							
DC-9	run 4	J							
DC-9	run 5	J							
DC-9	run 6	J							
DC-9	run 7	J							
DC-8 Turbojet	run 1	J		PR					PR
DC-8 Turbojet	run 2	J		PR					
DC-8 Turbojet	run 3	J							
DC-8 Turbojet	run 4	J							
DC-8 Turbojet	run 5	J		PR	PR	PR		PR	PR
DC-8 Turbojet	run 6	J							
DC-8 Turbofan	run 1	J							PR
DC-8 Turbofan	run 2	J		PR		TR			
DC-8 Turbofan	run 3	J							
DC-8 Turbofan	run 4	J							
DC-8 Turbofan	run 5	J							
DC-8 Turbofan	run 6	J		PR		PR			
DC-8 Turbofan	run 7	J	PR	PR	PR	PR		PR	HE
747	run 1	J							
747	run 2	J				WT		TR	PR
747	run 3	J							
747	run 4	J							
747	run 5	J							
747	run 6	J							
747	run 7	J							
L-1011	run 1	J							
L-1011	run 2	J							
L-1011	run 3	J							
L-1011	run 4	J							HE
L-1011	run 5	J		PR		PR			PR
L-1011	run 6	J							
DC-10	run 1	J			HE	TR	WT		
DC-10	run 2	J			HE	TR		TR	
DC-10	run 3	J							HE
DC-10	run 4	J			HE	TR	TR	TR	
DC-10	run 5	J							
DC-10	run 6	J							
DC-10	run 7	J							
DC-10	run 8	J							PR
DC-10	run 9	J							HE
DC-10	run 10	J							HE
DC-10	run 11	J							

C.1.3 Test Results - Run 3

			Number of Features						
Source		Class	4	5	6	7	8	9	10
T-38	run 1	J	PR	PR					
T-38	run 2	J		PR					
T-38	run 3	J	PR	PR				PR	
T-38	run 4	J							
T-38	run 5	J		PR		PR			
T-38	run 6	J		PR					
T-38	run 7	J		PR					
T-38	run 8	J							
Wind Turbine	run 1	WT		HE	HE	HE		HE	
Wind Turbine	run 2	WT	HE	HE	HE	HE		HE	
Wind Turbine	run 3	WT		HE	HE	HE		HE	
Wind Turbine	run 4	WT		HE	HE	HE		HE	
Wind Turbine	run 5	WT	HE	HE	HE	HE		HE	
Wind Turbine	run 6	WT	HE	HE	TR	TR		HE	
Wind Turbine	run 7	WT			HE	HE			
Wind Turbine	run 8	WT		HE	HE	HE		HE	
Wind Turbine	run 9	WT		HE	HE	HE		HE	
Wind Turbine	run 10	WT		HE	HE	HE		HE	
Wind Turbine	run 11	WT		HE	HE	HE		HE	
Wind Turbine	run 12	WT		HE	HE	HE		HE	
Wind Turbine	run 13	WT		HE	HE	HE		HE	
Wind Turbine	run 14	WT		TR	HE	HE		HE	
Wind Turbine	run 15	WT		HE	HE	HE		HE	
Wind Turbine	run 16	WT		HE	HE	HE		HE	
Wind Turbine	run 17	WT	HE	HE	HE	HE		HE	
Wind Turbine	run 18	WT		HE	HE	HE		HE	
Wind Turbine	run 19	WT		HE	HE	HE		HE	
Wind Turbine	run 20	WT		HE	HE	HE		HE	
Helicopter	run 1	HE							
Helicopter	run 2	HE							
Helicopter	run 3	HE							
Helicopter	run 4	HE				PR			
Helicopter	run 5	HE							
Helicopter	run 6	HE							
Helicopter	run 7	HE							
Helicopter	run 8	HE					TR	PR	
Helicopter	run 9	HE						PR	

Number of Features									
Source		Class	4	5	6	7	8	9	10
Helicopter	run 10	HE							
Helicopter	run 11	HE							
Helicopter	run 12	HE							
Helicopter	run 13	HE							
Helicopter	run 14	HE							
Train	run 1	TR							
Train	run 2	TR							
Train	run 3	TR							
Train	run 4	TR							
Train	run 5	TR							
Train	run 6	TR	HE			HE	HE	HE	WT
Train	run 7	TR	HE						
Train	run 8	TR						PR	
Train	run 9	TR	HE						
Train	run 10	TR							
Train	run 11	TR							
Train	run 12	TR							
Train	run 13	TR							
Train	run 14	TR							
Train	run 15	TR							
Train	run 16	TR	HE						WT
Train	run 17	TR	HE				HE	HE	WT
Train	run 18	TR							WT
King Air		PR							
Nord 262		PR		TR					
P-3		PR							
EMB-110		PR							
Dash 7		PR							
B-727		J	PR	PR					
DC-9		J	PR		PR	PR	PR	PR	PR
Viscount		PR							
Shorts 330		PR	HE						
YS-11		PR							
C-130		PR							
Swearingen Metro		PR	J	J	J		J	J	
B-707		J	PR	PR	PR	PR	PR	PR	PR
DC-10		J	PR		PR	PR			
A-300		J	PR	PR	PR				
T-28		PR		TR					
King Air		PR	TR	HE	HE			HE	
Nord 262		PR							
P-3		PR	HE			J			
EMB-110		PR							
Dash 7		PR		TR					
B-727		J	PR	PR	PR		PR	PR	PR

Number of Features									
Source		Class	4	5	6	7	8	9	10
DC-9		J	PR	PR	PR		PR	PR	PR
Viscount		PR	TR	TR					
Viscount		PR	TR	TR			TR		
Shorts 330		PR	HE			TR	J		
YS-11		PR	TR	TR					
C-130		PR							
Swearingen Metro		PR							
B-707		J	PR	PR	PR	PR	PR	PR	PR
DC-10		J	PR	PR	PR		PR	PR	PR
A-300		J	PR	PR	PR		PR	PR	PR
MU-2		PR							
Cessna 152		PR			J				
Cessna 172		PR							
Supercub		PR							
Bonanza IV		PR							
G. Tiger		PR							
727-100	run 1	J							
727-100	run 2	J	PR						
727-100	run 3	J							
727-100	run 4	J							
727-100	run 5	J							
727-100	run 6	J							
727-100	run 7	J							
727-100	run 8	J							
727-100	run 9	J	PR						
727-100	run 10	J							
727-100	run 11	J							
727-100	run 12	J	PR	PR					
727-100	run 13	J	PR						
727-200	run 1	J							
727-200	run 2	J							
727-200	run 3	J							
727-200	run 4	J							
727-200	run 5	J							
727-200	run 6	J							
727-200	run 7	J							
727-200	run 8	J							
727-200	run 9	J					PR		
727-200	run 10	J							
737	run 1	J							
737	run 2	J				PR			
737	run 3	J							
737	run 4	J	PR						
737	run 5	J							
737	run 6	J							

Number of Features									
Source		Class	4	5	6	7	8	9	10
DC-9	run 1	J							
DC-9	run 2	J							
DC-9	run 3	J							
DC-9	run 4	J							
DC-9	run 5	J							
DC-9	run 6	J							
DC-9	run 7	J							
DC-8 Turbojet	run 1	J	PR				PR		
DC-8 Turbojet	run 2	J	PR						
DC-8 Turbojet	run 3	J	PR						
DC-8 Turbojet	run 4	J	PR						
DC-8 Turbojet	run 5	J			PR		PR	PR	PR
DC-8 Turbojet	run 6	J							
DC-8 Turbofan	run 1	J			PR	TR		PR	PR
DC-8 Turbofan	run 2	J	PR		PR			PR	PR
DC-8 Turbofan	run 3	J	PR						PR
DC-8 Turbofan	run 4	J	PR						
DC-8 Turbofan	run 5	J	PR						
DC-8 Turbofan	run 6	J	PR	PR	PR	TR	PR	PR	PR
DC-8 Turbofan	run 7	J	PR				PR		PR
747	run 1	J	PR			PR		PR	
747	run 2	J	TR	PR	PR	PR	PR	PR	PR
747	run 3	J	PR						
747	run 4	J	PR	PR				PR	
747	run 5	J	PR					PR	
747	run 6	J	HE	PR	PR	HE	PR	PR	PR
747	run 7	J	PR	PR	PR	PR	PR	PR	PR
L-1011	run 1	J							
L-1011	run 2	J							
L-1011	run 3	J	PR						
L-1011	run 4	J							
L-1011	run 5	J		PR	PR		PR	PR	PR
L-1011	run 6	J	PR						PR
DC-10	run 1	J			HE	PR			
DC-10	run 2	J			PR	PR			
DC-10	run 3	J	PR	PR	PR	PR		PR	
DC-10	run 4	J	PR	PR	PR	PR			
DC-10	run 5	J							
DC-10	run 6	J			PR			PR	
DC-10	run 7	J							
DC-10	run 8	J	PR		PR	PR		PR	
DC-10	run 9	J							
DC-10	run 10	J	PR	PR	PR	PR	PR	PR	PR
DC-10	run 11	J	PR	PR	PR	PR	PR	PR	PR
DC-10	run 12	J		PR	PR	PR	PR	PR	PR
DC-10	run 13	J	HE	PR	PR	HE	PR	PR	PR

C.1.4 Test Results - Run 4

			Number of Features						
Source		Class	4	5	6	7	8	9	10
T-38	run 1	J							
T-38	run 2	J		PR					
T-38	run 3	J							
T-38	run 4	J							
T-38	run 5	J		PR					
T-38	run 6	J							
T-38	run 7	J							
T-38	run 8	J							
Wind Turbine	run 1	WT							
Wind Turbine	run 2	WT							
Wind Turbine	run 3	WT	TR					HE	HE
Wind Turbine	run 4	WT						HE	
Wind Turbine	run 5	WT						HE	
Wind Turbine	run 6	WT						HE	
Wind Turbine	run 7	WT		TR		TR	TR	TR	TR
Wind Turbine	run 8	WT							
Wind Turbine	run 9	WT				TR		TR	
Wind Turbine	run 10	WT		TR				TR	TR
Wind Turbine	run 11	WT		TR		TR			
Wind Turbine	run 12	WT		TR					TR
Wind Turbine	run 13	WT		TR		TR		TR	TR
Wind Turbine	run 14	WT						HE	
Wind Turbine	run 15	WT						HE	
Wind Turbine	run 16	WT							
Wind Turbine	run 17	WT							
Wind Turbine	run 18	WT							
Wind Turbine	run 19	WT							
Wind Turbine	run 20	WT							
Helicopter	run 1	HE		TR		TR		TR	
Helicopter	run 2	HE							
Helicopter	run 3	HE			J		WT	WT	
Helicopter	run 4	HE							
Helicopter	run 5	HE							
Helicopter	run 6	HE							
Helicopter	run 7	HE			WT				
Helicopter	run 8	HE			J				
Helicopter	run 9	HE							

Number of Features									
Source		Class	4	5	6	7	8	9	10
Helicopter	run 10	HE							
Helicopter	run 11	HE							
Helicopter	run 12	HE							
Helicopter	run 13	HE							
Helicopter	run 14	HE							
Train	run 1	TR							
Train	run 2	TR							
Train	run 3	TR							
Train	run 4	TR		HE	WT		WT		
Train	run 5	TR							
Train	run 6	TR	WT						
Train	run 7	TR					WT		
Train	run 8	TR							
Train	run 9	TR					WT		
Train	run 10	TR							
Train	run 11	TR							
Train	run 12	TR							
Train	run 13	TR							
Train	run 14	TR							
Train	run 15	TR							
Train	run 16	TR	WT		WT				
Train	run 17	TR							
Train	run 18	TR							
King Air		PR							
Nord 262		PR		TR					J
P-3		PR							
EMB-110		PR							
Dash 7		PR							
B-727		J			PR				PR
DC-9		J				PR			
Viscount		PR						J	
Shorts 330		PR							
YS-11		PR							
C-130		PR							
Swearingen Metro		PR					J		
B-707		J	PR	PR	PR	PR	PR		
DC-10		J							
A-300		J						PR	
T-28		PR	J	J	J	J	J	J	J
King Air		PR	J	TR	J			J	J
Nord 262		PR		TR		J		J	J
P-3		PR	J		J	J	HE	J	J
EMB-110		PR							
Dash 7		PR		J		J		J	J
B-727		J	PR		PR			PR	PR

Number of Features									
Source		Class	4	5	6	7	8	9	10
DC-9		J	PR	TR	PR	PR	PR	PR	PR
Viscount		PR	J	TR		TR			J
Viscount		PR	J	TR	J	J		J	J
Shorts 330		PR	J	J	J	J	J	J	J
YS-11		PR	J	TR					
C-130		PR	J			J			
Swearingen Metro		PR	J	J	J	J	J	J	J
B-707		J	PR	PR	PR	PR	PR	PR	PR
DC-10		J	PR	PR	PR		PR		PR
A-300		J	PR	PR	PR	PR	PR		
MU-2		PR							
Cessna 152		PR							
Cessna 172		PR					HE		
Supercub		PR							
Bonanza IV		PR							
G. Tiger		PR							
727-100	run 1	J							
727-100	run 2	J							
727-100	run 3	J							
727-100	run 4	J							
727-100	run 5	J							
727-100	run 6	J							
727-100	run 7	J							
727-100	run 8	J							
727-100	run 9	J							
727-100	run 10	J							
727-100	run 11	J							
727-100	run 12	J							
727-100	run 13	J							
727-200	run 1	J							
727-200	run 2	J							
727-200	run 3	J							
727-200	run 4	J							
727-200	run 5	J							
727-200	run 6	J							
727-200	run 7	J							
727-200	run 8	J							
727-200	run 9	J							
727-200	run 10	J							
737	run 1	J	PR		PR				
737	run 2	J	PR		PR				
737	run 3	J							
737	run 4	J							
737	run 5	J							
737	run 6	J							

Number of Features									
Source		Class	4	5	6	7	8	9	10
DC-9	run 1	J							
DC-9	run 2	J							
DC-9	run 3	J							
DC-9	run 4	J							
DC-9	run 5	J							
DC-9	run 6	J							
DC-9	run 7	J							
DC-8 Turbojet	run 1	J							
DC-8 Turbojet	run 2	J							
DC-8 Turbojet	run 3	J							
DC-8 Turbojet	run 4	J							
DC-8 Turbojet	run 5	J	PR	PR			PR		
DC-8 Turbojet	run 6	J							
DC-8 Turbofan	run 1	J							
DC-8 Turbofan	run 2	J							
DC-8 Turbofan	run 3	J							
DC-8 Turbofan	run 4	J							
DC-8 Turbofan	run 5	J							
DC-8 Turbofan	run 6	J		PR					PR
DC-8 Turbofan	run 7	J	PR	PR	PR	PR	PR		PR
747	run 1	J							
747	run 2	J							
747	run 3	J							
747	run 4	J							
747	run 5	J							
747	run 6	J		PR			TR		
747	run 7	J							
L-1011	run 1	J		HE			WT		
L-1011	run 2	J	TR	HE	TR		WT		TR
L-1011	run 3	J							
L-1011	run 4	J							
L-1011	run 5	J	PR	PR			PR		PR
L-1011	run 6	J							
DC-10	run 1	J	PR						PR
DC-10	run 2	J		PR			PR		PR
DC-10	run 3	J	PR	PR		PR	PR		PR
DC-10	run 4	J	TR	PR	TR	PR	PR		PR
DC-10	run 5	J							
DC-10	run 6	J							
DC-10	run 7	J							
DC-10	run 8	J							
DC-10	run 9	J							
DC-10	run 10	J		PR			PR		PR
DC-10	run 11	J	PR	PR	TR	PR	PR	TR	PR
DC-10	run 12	J							
DC-10	run 13	J	TR	HE			WT		

C.2 Recognition Rates

This section contains the recognition rates of the different classifiers, calculated from the output given in appendix C.1. The values are all given as percent of sources correctly identified, for each classifier, number of features, and type of source.

Overall Recognition Rates (percent)							
	Number of Features						
Run #	4	5	6	7	8	9	10
1	85.6	85.0	87.9	79.2	87.9	84.4	85.6
2	88.9	84.2	80.1	81.3	90.1	81.9	81.9
3	65.9	68.8	71.1	74.0	85.0	69.9	85.0
4	83.2	78.6	86.1	86.7	83.8	85.0	82.7
Jet Plane Recognition Rates (percent)							
	Number of Features						
Run #	4	5	6	7	8	9	10
1	95.7	90.2	81.5	91.3	94.6	85.9	92.4
2	95.6	84.6	90.1	87.9	89.0	85.7	85.7
3	54.8	71.0	71.0	77.4	78.5	72.0	76.3
4	81.7	78.5	86.0	90.3	81.7	94.6	83.9
Propeller Plane Recognition Rates (percent)							
	Number of Features						
Run #	4	5	6	7	8	9	10
1	89.7	72.4	96.6	62.1	72.4	75.9	65.5
2	57.1	78.6	57.1	60.7	85.7	67.9	57.1
3	71.4	71.4	89.3	92.9	89.3	92.9	100.0
4	67.9	64.3	78.6	67.9	78.6	67.9	64.3

Wind Turbine Recognition Rates (percent)							
	Number of Features						
Run #	4	5	6	7	8	9	10
1	10.0	90.0	85.0	20.0	65.0	85.0	80.0
2	100.0	100.0	55.0	100.0	100.0	55.0	100.0
3	80.0	5.0	0.0	0.0	100.0	5.0	100.0
4	95.0	75.0	100.0	80.0	95.0	50.0	75.0
Helicopter Recognition Rates (percent)							
	Number of Features						
Run #	4	5	6	7	8	9	10
1	100.0	92.9	100.0	92.9	100.0	100.0	71.4
2	78.6	71.4	78.6	42.9	100.0	100.0	57.1
3	100.0	100.0	100.0	92.9	92.9	85.7	100.0
4	100.0	92.9	78.6	92.9	92.9	85.7	100.0
Train Recognition Rates (percent)							
	Number of Features						
Run #	4	5	6	7	8	9	10
1	100.0	66.7	100.0	100.0	94.4	77.8	100.0
2	100.0	83.3	94.4	88.9	83.3	100.0	100.0
3	72.2	100.0	100.0	94.4	88.9	83.3	77.8
4	88.9	94.4	88.9	100.0	83.3	100.0	100.0

Appendix D. FORTRAN Program Listings

The FORTRAN program listings for the complete classification system are given in this section. Three sets of programs were used in training and testing the system, *Trun*, *Fcals*, and *CEO*. Fcals was used to generate the feature vectors for the training and evaluation patterns, from stored PSD and autocorrelation values for each source recording. CEO controlled the training and feature selection with the pattern vectors generated by Fcals.

Trun was a type of executive program used in the testing mode of operation. Trun controlled data acquisition, and the calculation of the PSD, autocorrelation, and features. In addition, Trun managed the retrieval of classifiers for the tree classification structures. Each classifier was called up according to the results of the preceding level, until the bottom level was reached, at which point classification ended.

All of the programs were written in FORTRAN. The training programs manipulated the pattern vectors in a two dimensional array. A typical array of training patterns contained the identification of a given a pattern as the first column entry, and the feature values as subsequent entries. Different patterns were located on different rows in the array. As a classifier was being trained, the row elements of the matrices were fed to the perceptron algorithm one by one until they were all classified correctly.

D.1 Fcals Listing

Fcals was used to calculate the feature values. The PSD and autocorrelation were read in, then the specifications for the features were read in. The features were then calculated according to the specifications, and assembled into training or evaluation sets as indicated by the user. These sets were then stored for later use in training the classifiers.

A similar routine was called by Trun during testing, except once all the features were calculated, the ones corresponding to the best set found during training were extracted.

```
C-------------------------------------------------------
C     FEATURE CALCULATOR
C
C     READS IN PARAMETERS FROM DATA FILE IN THE FOLLOWING
C     ORDER:
C         NPPDF - NUMBER OF POINTS IN PDF
C         DELF - DELTA FREQUENCY IN PDF
C         NPKDET - NUMBER OF TIMES PEAKDET IS TO BE CALLED
C         NLFIL - LENGTH OF LONG FILTER FOR PEAKDET, IN ARRAY
C                 FORM IF PEAKDET TO BE CALLED MORE THAN ONCE
C         NSFIL - LENGTH OF SHORT FILTER FOR PEAKDET, IN ARRAY
C                 FORM IF PEAKDET TO BE CALLED MORE THAN ONCE
C         THRESH - THRESHOLD IN dB FOR PEAK DETECTION, IN ARRAY
C                FORM IF PEAKDET CALLLED MORE THEN ONCE
C         NSPSUM - NUMBER OF TIMES SPECSUM IS TO BE CALLED
C         FSTART - STARTING FREQUENCY FOR SPECSUM, IN ARRAY FORM
C              IF SPECSUM CALLED MORE THAN ONCE
C         FEND - ENDING FREQUENCY FOR SPECSUM, IN ARRAY FORM
C                IF SPECSUM CALLED MORE THAN ONCE
C         NBANDS - NUMBER OF BANDS FOR ENERGY CALC FOR SPECSUM, IN
C              ARRAY FORM IF SPECSUM CALLED MORE THAN ONCE
C         NRATIO - NUMBER OF POWER RATIOS TO CALCULATE
C         NBRAT - POWER RATIO NUMBERS TO COMPARE
C         NPOW - NUMBER OF ACCUMULATED POWER CALCS
C         ACCPOW - PERCENT ACCUMULATED POWER FOR CALCS
C         NPAUTO - NUMBER OF POINTS IN THE AUTO-CORR
C-------------------------------------------------------------
      INTEGER NLFIL(10),NSFIL(10),NBANDS(100),DESFEAT,
     1        NBRAT(100,100),NRATIO(100),NTBRAT(100)
      REAL FSTART(10),FEND(10),ACCPOW(100),FACCPOW(100),
     1     FEAT(200),PATS(200,200),BESTSET(30),PSD(2050),
     2     BLPERC(20),THRESH(10),CRNTFT(200),AUTOC(100),RATIO(50)

      LOGICAL BSET
      CHARACTER*7 XTYPE
      CHARACTER*35 PSDFILE,PATSFL,BSETFL,PARAMSFL,RUNFL
      CHARACTER*4 CHECK,AGAINST
      NPAT=0
      AGAINST='  AC'
C-----------------------------------------------------------------
C     READ IN PARAMETERS FOR FEATURE CALCULATIONS
C-----------------------------------------------------------------
      WRITE(*,*) 'ENTER NAME OF FILE WITH CALC PARAMETERS'
```

```
      READ(*,34) PARAMSFL
      OPEN(UNIT=15,FILE=PARAMSFL,STATUS='OLD')
      READ(15,*) NPPSD                        ! NUM POINTS IN PSD
      READ(15,*) DELF                         ! DELTA FREQUENCY
      READ(15,*) NPKDET                       ! TIMES TO RUN PEAKDET
      READ(15,*) (NLFIL(I), I=1,NPKDET)       ! LENGTHS OF LONG FILTER
      READ(15,*) (NSFIL(I), I=1,NPKDET)       ! LENGTHS OF SHORT FILTER
      READ(15,*) (THRESH(I), I=1,NPKDET)      ! PEAK THRESHOLDS
      READ(15,*) NSPSUM                       ! TIMES TO RUN SPECSUM
      READ(15,*) (FSTART(I), I=1,NSPSUM)      ! STARTING FREQUENCIES
      READ(15,*) (FEND(I), I=1,NSPSUM)        ! ENDING FREQUENCIES
      READ(15,*) (NBANDS(I), I=1,NSPSUM)      ! NUMBER OF BANDS
      READ(15,*) (NRATIO(I),I=1,NSPSUM)         ! NUMBER OF RATIOS
      DO 306 I=1,NSPSUM
        READ(15,*) (NBRAT(I,J),J=1,2*NRATIO(I))  ! RATIOS TO CALCULATE
 306  CONTINUE
      READ(15,*) NPOW                         ! NUMBER OF ACCUMULATED POWER
      READ(15,*) (ACCPOW(I), I=1,NPOW)        ! ACCU POWER POINTS
      READ(15,*) NPAUTO                       ! NUM POINTS IN AUTO-CORR
      CLOSE(UNIT=15)
C-- CALCULATE NUMBER OF FEATURES TO BE CALCULATED
      NCALFEAT=NPKDET                       ! FROM PEAKDET
      DO 98 I=1,NSPSUM                        ! FROM SPECSUM
         NCALFEAT=NCALFEAT+NBANDS(I)+NPOW+NRATIO(I)
 98   CONTINUE
      NCALFEAT=NCALFEAT+2                     ! FROM AUTOCALS
      WRITE(*,*) 'NCALFEAT = ',NCALFEAT

      TYPE 10
 10   FORMAT(1X,'ENTER 1 IF PATTERN IS KNOWN, 0 IF UNKNOWN')
      READ(5,*) ITYPE
      WRITE(*,*) ITYPE
      IF (ITYPE.EQ.1) THEN
         XTYPE='KNOWN'
         WRITE(*,*) 'ENTER NAME OF FILE FOR TRAINING PATS'
         READ(*,34) PATSFL
 34      FORMAT(A35)
      ELSE
         XTYPE='UNKNOWN'
         WRITE(*,*) 'ENTER NAME OF FILE FOR UNKNOWN PATS'
         READ(*,34) PATSFL
      END IF
      TYPE 12,XTYPE
 12   FORMAT(1X,'ARE OTHER',A7,'PATTERNS ALREADY STORED(0=N,1=Y)')
      READ(5,*) IOTH
      WRITE(*,*) IOTH
      IF (IOTH.EQ.1) THEN              ! LOAD OTHER PATTERNS
          IF(ITYPE.EQ.1) THEN           ! LOAD KNOWNS
             OPEN(UNIT=9,FILE=PATSFL,STATUS='UNKNOWN')
                READ(9,*) NPAT,NCLASS,NFEAT
             READ(9,*) ((PATS(I,J), J=1,NFEAT+1), I=1,NPAT)
             CLOSE(UNIT=9)
          ELSE
             OPEN(UNIT=11,FILE=PATSFL,STATUS='UNKNOWN')
             READ(11,*) NPAT,NCLASS,NFEAT    ! LOAD UNKNOWNS
             READ(11,*) ((PATS(I,J), J=1,NFEAT), I=1,NPAT)
             CLOSE(UNIT=11)
          END IF
          IF (NFEAT.NE.NCALFEAT) THEN
             WRITE(*,*) 'ENTER NAME OF FILE CONTAINING BESTSET FEATS'
             READ(*,34) BSETFL
             OPEN(UNIT=13,FILE=BSETFL,STATUS='UNKNOWN')
             BSET=.TRUE.                                  ! ASSUME A BESTSE
             TYPE 13                                      ! EXISTS - LOAD N
 13          FORMAT(1X,'LOADING BESTSET FEATURE NUMBERS')
             READ(13,*) DESFEAT
             READ(13,*) (BESTSET(I),I=1,DESFEAT)
```

```
            CLOSE(UNIT=13)
         ELSE
            BSET=.FALSE.
         END IF
      ELSE
         BSET=.FALSE.
      END IF


C-----------------------------------------------------------------
C    BEGIN FEATURE CALS
C-----------------------------------------------------------------
 100  IFEAT=1
      TYPE 19
 19   FORMAT(1X,'ENTER PSD FILE NAME')
      READ(5,21) PSDFILE
 21   FORMAT(A24)
      WRITE(*,*) PSDFILE
      OPEN(UNIT=50,FILE=PSDFILE,STATUS='OLD')
 765  READ(50,764) CHECK
 764  FORMAT(A4)
      IF(CHECK.NE.AGAINST) GOTO 765
      READ(50,*) (AUTOC(I),I=1,NPAUTO)
      DO 766 I=1,4
         READ(50,*)
 766  CONTINUE
      READ(50,*) (PSD(I), I=1,NPPSD)       ! READ IN PSD
      CLOSE(UNIT=50)
      DO 821 I=1,NPPSD
         IF(PSD(I).LT.0.0) PSD(I)=0.0
 821  CONTINUE
      IF (ITYPE.EQ.1) THEN
         TYPE 11
 11      FORMAT(1X,'ENTER CLASS NUMBER OF PATTERN')
         READ(5,*) ICLASS
         WRITE(*,*) ICLASS
      END IF

C-----------------------------------------------------------------
C
C    CALL PEAK DETECTION ROUTINE
C
C-----------------------------------------------------------------
      DO 30 I=1,NPKDET
         CALL PEAKDET(PSD,NPPSD,NLFIL(I),NSFIL(I),THRESH(I),
     1                NPEAK,NHARM)
         FEAT(IFEAT)=NPEAK
         IFEAT=IFEAT+1
C         FEAT(IFEAT)=NHARM
C         IFEAT=IFEAT+1
 30   CONTINUE

C-----------------------------------------------------------------
C
C    CALL SPECTRAL SUM ROUTINE
C
C-----------------------------------------------------------------
      DO 40 I=1,NSPSUM
         DO 743 J=1,2*NRATIO(I)
            NTBRAT(J)=NBRAT(I,J)
 743     CONTINUE
         CALL SPECSUM(PSD,NPPSD,DELF,FSTART(I),FEND(I),
     1                NBANDS(I),NPOW,ACCPOW,FACCPOW,BLPERC,
     2                NRATIO(I),NTBRAT,RATIO)
         DO 50 J=1,NBANDS(I)
            FEAT(IFEAT)=BLPERC(J)*100.
              IFEAT=IFEAT+1
```

```
 50       CONTINUE
          DO 54 J=1,NRATIO(I)
             FEAT(IFEAT)=RATIO(J)*100.
             IFEAT=IFEAT+1
 54       CONTINUE
          DO 60 J=1,NPOW
             FEAT(IFEAT)=FACCPOW(J)*100.
             IFEAT=IFEAT+1
 60       CONTINUE
 40    CONTINUE

C-----------------------------------------------------------------
C
C     CALL AUTOCORRELATION ROUTINE
C
C-----------------------------------------------------------------
       STDDEVAC=0.0
       CALL AUTOCALS(AUTOC,NPAUTO,AVEAC,STDDEVAC)

       FEAT(IFEAT)=AVEAC*100.
       IFEAT=IFEAT+1
       FEAT(IFEAT)=STDDEVAC*100.
       IFEAT=IFEAT+1

C-- ALL FEATURE CALS OVER
C-- MAKE ROOM FOR CLASS ID IF A TRAINING PATTERN
       NPAT=NPAT+1
       IF (ITYPE.EQ.1) THEN
           DO 43 I=IFEAT,2,-1
              FEAT(I)=FEAT(I-1)
 43        CONTINUE
           FEAT(1)=ICLASS
       END IF
       IFEAT=IFEAT-1

C-----------------------------------------------------------------
C     TRANSFER FEATURES VALUES IF THERE IS A BESTSET
C-----------------------------------------------------------------
       IF (BSET) THEN
          CALL ZUT(FEAT,CRNTFT,IFEAT,BESTSET,ITYPE)
          DO 45 I=1,DESFEAT+1
             FEAT(I)=CRNTFT(I)
 45       CONTINUE
          IFEAT=DESFEAT
       END IF

C-----------------------------------------------------------------
C     PUT NEW VALUES INTO PATTERN MATRIX
C-----------------------------------------------------------------
       DO 55 I=1,IFEAT+ITYPE
          PATS(NPAT,I)=FEAT(I)
 55    CONTINUE
       TYPE 70,XTYPE
 70    FORMAT(1X,'ANY MORE',A10,'PATTERNS (0=N,1=Y)')
       READ(5,*) IANY
       WRITE(*,*) IANY
       IF (IANY.EQ.1) GOTO 100
C-----------------------------------------------------------
C     SAVE PATTERNS
C-----------------------------------------------------------
       IF (INPUT.EQ.0) THEN
          CLOSE(UNIT=55)
       END IF
       IF (ITYPE.EQ.1) THEN
          REWIND 9
          OPEN(UNIT=9,FILE=PATSFL,STATUS='UNKNOWN')
          WRITE(9,*) NPAT,NCLASS,IFEAT
```

```
         DO 110 I=1,NPAT
            WRITE(9,*) (PATS(I,J),J=1,IFEAT+1)
110      CONTINUE
         CLOSE(UNIT=9)
      ELSE
         REWIND 11
         OPEN(UNIT=11,FILE=PATSFL,STATUS='UNKNOWN')
         WRITE(11,*) NPAT,NCLASS,IFEAT
         DO 120 I=1,NPAT
            WRITE(11,*) (PATS(I,J),J=1,IFEAT)
120      CONTINUE
         CLOSE(UNIT=11)
      END IF
      STOP
      END
```

D.2 Trun Listing

Trun was used during testing to manage the classification of an unknown pattern. Trun supervised the calculation of the PSD, autocorrelation, and the features, then extracted the features selected by the feature selection routine. The relevant classifiers were called up for classifying the pattern vector, and Trun stopped when the unknown had been classified.

Trun used many of the same subroutines as the training programs because all patterns required the PSD and autocorrelation as well as feature calculations.

```
C-----------------------------------------------------------------
C     PROGRAM TO EXECUTE THE PATTERN RECOG SYSTEM, CONSISTING
C     OF TSNAP, TCLASS, AND TCALS.
C     AFTER ACQUISITION, TSNAP IS USED TO CALCULATE THE PSD AND AUTOC.
C           TCALS THEN CALCULATES FEATURE VALUES
C           TCLASS EVALUATES THE DECISION FUNCTIONS
C
C
C-----------------------------------------------------------------
      INTEGER IBUFF(65536),BESTSET(30),CC,L(5),DESFEAT,F(4)
      REAL W(20,200),PSD(2049),AUTOC(60),D(20),CU(200),
     1     FEAT(200)
      LOGICAL*1 WTFL(11)
      LOGICAL FOUND
      CHARACTER*20 C(10)

      PARAMETER(N=2047,IREPCOUNT=31)

      DATA IO,LEVEL,L(1),L(2),L(3),L(4)/0,1,0,0,0,0/

      WRITE(*,*) 'ENTER NUMBER OF FEATURES FOR LEVEL 1(<10)'
      READ(*,*) F(1)
      WRITE(*,*) 'ENTER NUMBER OF FEATURES FOR LEVEL 2(<10)'
      READ(*,*) F(2)
      WRITE(*,*) 'ENTER NUMBER OF FEATURES FOR LEVEL 3(<10)'
      READ(*,*) F(3)
      WRITE(*,*) 'ENTER NUMBER OF FEATURES FOR LEVEL 4(<10)'
      READ(*,*) F(4)
```

```
C-------------------------------------------------------------
C    GET TIME HISTORY HERE
C-------------------------------------------------------------
      write(*,*)
      write(*,*) 'Data Acquisition -'
      write(*,*)
      CALL GET_1CHAN_DATA(IBUFF,N,IREPCOUNT)

C-------------------------------------------------------------
C    CALCULATE PSD AND AUTOCORRELATION
C-------------------------------------------------------------
      write(*,*) 'PSD and Autocorrelation Calculation -'
      write(*,*)
      CALL TSNAP(IBUFF,PSD,AUTOC)

C-------------------------------------------------------------
C    CALCULATE FEATURES
C-------------------------------------------------------------
      write(*,*) 'Feature Calculation -'
      write(*,*)
      CALL TCALS(PSD,AUTOC,CU,LN)
C-------------------------------------------------------------
C    CLASSIFY SIGNAL USING WEIGHT MATRIX,CALL TZUT IF NECESSARY
C-------------------------------------------------------------
      write(*,*) 'Signal Classification :'
      write(*,*)
 35   ENCODE(11,300,WTFL)F(LEVEL),L(1),L(2),L(3),L(4),IO
 300  FORMAT('W',5I1.1,'.DAT',A1)
      INQUIRE(FILE=WTFL,EXIST=FOUND)
      IF(.NOT.FOUND) GOTO 500
 305  CALL ASSIGN(12,WTFL,11)
      READ(12,*) IRID,DESFEAT,M
      READ(12,14) (C(I),I=1,M)
 14   FORMAT(A20)
      READ(12,*) (BESTSET(I),I=1,DESFEAT)
      DO 15 I=1,M
         READ(12,*) (W(I,J),J=1,DESFEAT+1)
 15   CONTINUE

      DO 90 J=1,DESFEAT
         FEAT(J)=CU(BESTSET(J))
 90   CONTINUE

      CALL TCLASS(FEAT,DESFEAT,W,M,CC)
      WRITE(*,138) C(CC)
 138  FORMAT(1X,A20)
      L(LEVEL)=CC
      LEVEL=LEVEL+1
      GOTO 35

 500  write(*,*)
      write(*,*)
      WRITE(*,510) WTFL
 510  FORMAT(1X,'CLASSIFICATION OVER; ',11A1,' NOT FOUND.')
      STOP
      END
```

D.2.1 Subroutine Tcals

Tcals was used to calculate the feature values by Trun once the PSD and autocorrelation had been

calculated. Tcals is similar to Fcals except Tcals is arranged to calculate feature values for only one pattern at a time.

```
C-------------------------------------------------------
C     FEATURE CALCULATOR
C
C     READS IN PARAMETERS IN THE
C     FOLLOWING ORDER:
C         NPPDF - NUMBER OF POINTS IN PDF
C         DELF - DELTA FREQUENCY IN PDF
C         NPKDET - NUMBER OF TIMES PEAKDET IS TO BE CALLED
C         NLFIL - LENGTH OF LONG FILTER FOR PEAKDET, IN ARRAY
C                 FORM IF PEAKDET TO BE CALLED MORE THAN ONCE
C         NSFIL - LENGTH OF SHORT FILTER FOR PEAKDET, IN ARRAY
C                 FORM IF PEAKDET TO BE CALLED MORE THAN ONCE
C         THRESH - THRESHOLD IN dB FOR PEAK DETECTION, IN ARRAY
C                FORM IF PEAKDET CALLLED MORE THEN ONCE
C         NSPSUM - NUMBER OF TIMES SPECSUM IS TO BE CALLED
C         FSTART - STARTING FREQUENCY FOR SPECSUM, IN ARRAY FORM
C              IF SPECSUM CALLED MORE THAN ONCE
C         FEND - ENDING FREQUENCY FOR SPECSUM, IN ARRAY FORM
C                IF SPECSUM CALLED MORE THAN ONCE
C         NBANDS - NUMBER OF BANDS FOR ENERGY CALC FOR SPECSUM, IN
C              ARRAY FORM IF SPECSUM CALLED MORE THAN ONCE
C         NRATIO - NUMBER OF POWER RATIOS TO CALCULATE
C         NBRAT - POWER RATIO NUMBERS TO COMPARE
C         NPOW - NUMBER OF ACCUMULATED POWER CALCS
C         ACCPOW - PERCENT ACCUMULATED POWER FOR CALCS
C         NPAUTO - NUMBER OF POINTS IN THE AUTO-CORR
C-------------------------------------------------------------
      SUBROUTINE TCALS(PSD,AUTOC,FEAT,NCALFEAT)
      INTEGER NLFIL(10),NSFIL(10),NBANDS(100),
     1        NBRAT(100,100),NRATIO(100),NTBRAT(100)
      REAL FSTART(10),FEND(10),ACCPOW(100),FACCPOW(100),
     1     FEAT(200),PSD(2049),
     2     BLPERC(20),THRESH(10),AUTOC(60),RATIO(50)

C-------------------------------------------------------------
C     READ IN PARAMETERS FOR FEATURE CALCULATIONS
C        FILL IN THE NAME OF THE PARAMETER FILE
C     OR SET THE DESIRED VALUES IN A DATA STATEMENT, FOR FASTER
C        OPERATION
C-------------------------------------------------------------
      OPEN(UNIT=15,FILE='PARAMS.DAT',STATUS='OLD')
        READ(15,*) NPPSD                    ! NUM POINTS IN PSD
        READ(15,*) DELF                     ! DELTA FREQUENCY
        READ(15,*) NPKDET                   ! TIMES TO RUN PEAKDET
        READ(15,*) (NLFIL(I), I=1,NPKDET)   ! LENGTHS OF LONG FILTER
        READ(15,*) (NSFIL(I), I=1,NPKDET)   ! LENGTHS OF SHORT FILTER
        READ(15,*) (THRESH(I), I=1,NPKDET)  ! PEAK THRESHOLDS
        READ(15,*) NSPSUM                   ! TIMES TO RUN SPECSUM
        READ(15,*) (FSTART(I), I=1,NSPSUM)  ! STARTING FREQUENCIES
        READ(15,*) (FEND(I), I=1,NSPSUM)    ! ENDING FREQUENCIES
        READ(15,*) (NBANDS(I), I=1,NSPSUM)  ! NUMBER OF BANDS
        READ(15,*) (NRATIO(I),I=1,NSPSUM)     ! NUMBER OF RATIOS
        DO 306 I=1,NSPSUM
          READ(15,*) (NBRAT(I,J),J=1,2*NRATIO(I))  ! RATIOS TO CALCULAT
 306    CONTINUE
        READ(15,*) NPOW                     ! NUMBER OF ACCUMULATED POW
        READ(15,*) (ACCPOW(I), I=1,NPOW)    ! ACCU POWER POINTS
        READ(15,*) NPAUTO                   ! NUM POINTS IN AUTO-CORR
      CLOSE(UNIT=15)
C-- CALCULATE NUMBER OF FEATURES TO BE CALCULATED
      NPAT=0
      NCALFEAT=NPKDET                    ! FROM PEAKDET
```

```
      DO 98 I=1,NSPSUM                    ! FROM SPECSUM
         NCALFEAT=NCALFEAT+NBANDS(I)+NPOW+NRATIO(I)
 98   CONTINUE
      NCALFEAT=NCALFEAT+2                 ! FROM AUTOCALS

C-------------------------------------------------------------
C     BEGIN FEATURE CALS
C     ASSUMING PSD AND AUTO CORR SENT DOWN IN MATRICES CALLLED
C     PSD AND AUTOC
C-------------------------------------------------------------
 100  IFEAT=1
C-------------------------------------------------------------
C
C     CALL PEAK DETECTION ROUTINE
C
C-------------------------------------------------------------
      DO 30 I=1,NPKDET
         CALL PEAKDET(PSD,NPPSD,NLFIL(I),NSFIL(I),THRESH(I),
     1                NPEAK,NHARM)
         FEAT(IFEAT)=NPEAK
         IFEAT=IFEAT+1
 30   CONTINUE
C-------------------------------------------------------------
C
C     CALL SPECTRAL SUM ROUTINE
C
C-------------------------------------------------------------
      DO 40 I=1,NSPSUM
         DO 743 J=1,2*NRATIO(I)
            NTBRAT(J)=NBRAT(I,J)
 743     CONTINUE
         CALL SPECSUM(PSD,NPPSD,DELF,FSTART(I),FEND(I),
     1                NBANDS(I),NPOW,ACCPOW,FACCPOW,BLPERC,
     2                NRATIO(I),NTBRAT,RATIO)
         DO 50 J=1,NBANDS(I)
            FEAT(IFEAT)=BLPERC(J)*100.
              IFEAT=IFEAT+1
 50      CONTINUE
         DO 54 J=1,NRATIO(I)
            FEAT(IFEAT)=RATIO(J)*100.
            IFEAT=IFEAT+1
 54      CONTINUE
         DO 60 J=1,NPOW
            FEAT(IFEAT)=FACCPOW(J)*100.
            IFEAT=IFEAT+1
 60      CONTINUE
 40   CONTINUE
C-------------------------------------------------------------
C
C     CALL AUTOCORRELATION ROUTINE
C
C-------------------------------------------------------------
      STDDEVAC=0.0
      CALL AUTOCALS(AUTOC,NPAUTO,AVEAC,STDDEVAC)

      FEAT(IFEAT)=AVEAC*100.
      IFEAT=IFEAT+1
      FEAT(IFEAT)=STDDEVAC*100.
      IFEAT=IFEAT+1

C-- ALL FEATURE CALS OVER
      NPAT=NPAT+1
      IFEAT=IFEAT-1
      RETURN
      END
```

D.2.1.1 Peakdet Listing

```
C-------------------------------------------------------------
C
C     PEAK DETECTION PROGRAM
C     TO BE USED ON PSD VALUES IN dB
C     VARIABLES:
C         NLFIL - LENGTH OF LONG FILTER
C         NSFIL - LENGTH OF SHORT FILTER
C         AVSFIL - SHORT FILTER AVERAGE
C         AVLFIL - LONG FILTER AVERAGE
C         THRESH - THRESHOLD IN dB FOR PEAK DETECTION
C         NPEAK - NUMBER OF PEAKS DETECTED
C         PEAKS - MATRIX CONTAINING STARTING AND ENDING
C                         VALUES OF PEAKS DETECTED
C-------------------------------------------------------------
      SUBROUTINE PEAKDET(PSD,NP,NLFIL,NSFIL,THRESH,
     1                    NPEAK,NHARM)
      INTEGER NP,NLFIL,NSFIL,PEAKS(200,2)
      REAL ADJUST(2050),PSD(2049),THRESH,AVEPEAK(200)

      LOGICAL DOWNPEAK,HARMFLAG
      NPEAK=0
      AVSFIL=0.0
      AVLFIL=0.0

      DO 10 I=NP,(NP-NLFIL),-1
         adjust(i)=psd(i)
         AVLFIL=AVLFIL+PSD(I)/FLOAT(NLFIL)
 10   CONTINUE
      DO 20 I=(NP-NLFIL-1),(NP-NLFIL-NSFIL),-1
         adjust(i)=psd(i)
         AVSFIL=AVSFIL+PSD(I)/FLOAT(NSFIL)
 20   CONTINUE
      DOWNPEAK=.TRUE.
      FMARK=I
 100  AVSFIL=AVSFIL-((PSD(FMARK+NSFIL)-PSD(FMARK))/FLOAT(NSFIL))
      IF((AVSFIL-AVLFIL).GE.THRESH) THEN
         ADJUST(FMARK)=AVLFIL
         IF(DOWNPEAK) THEN
            IF(FMARK.LE.1350) THEN
               NPEAK=NPEAK+1
               PEAKS(NPEAK,2)=FMARK
               DOWNPEAK=.FALSE.
            END IF
         END IF
      ELSE
         ADJUST(FMARK)=PSD(FMARK)
         IF(DOWNPEAK) GOTO 200
         PEAKS(NPEAK,1)=FMARK
         DOWNPEAK=.TRUE.
 200  END IF
      AVLFIL=AVLFIL-((ADJUST(FMARK+NLFIL+NSFIL)-ADJUST(FMARK))/
     1       FLOAT(NLFIL))
      FMARK=FMARK-1
      IF(FMARK.GT.0) GOTO 100
      RETURN
      END
```

D.2.1.2 Specsum Listing

```
C-------------------------------------------------------------
C
C     PROGRAM TO CALCULATE THE PARTIAL POWERS IN FREQUENCY
C     BANDS
C
C     IPOW COUNTS THE NUMBER OF ACCUMUALATED POWER CALCS,
```

```
C     AND WHEN IT EQUALS NPOW THE CALCS ARE DONE.
C-------------------------------------------------------------
      SUBROUTINE SPECSUM(PSD,NP,DELF,FSTART,FEND,NBANDS,
     1                   NPOW,ACCPOW,FACCPOW,BLPERC,
     2                   NRATIO,NBRAT,RATIO)
      INTEGER NBRAT(100)
      REAL ACCPOW(100),FACCPOW(100),BLSUM(2050),DFEN(2050),
     1        BLPERC(2050),PSD(2049),RATIO(50)

      NSTART=(FSTART/DELF+.5)
      NEND=(FEND/DELF+.5)
      NPPBL=FLOAT(NEND-NSTART)/FLOAT(NBANDS)+.5

      TOTEN=0.0
      IBLOCK=1
      BLSUM(1)=0.0
      DO 10 I=1,NP
C          DFEN(I)=(10.**(PSD(I)/10.)+10**(PSD(I+1)/10.))/2.
         DFEN(I)=(PSD(I)+PSD(I+1))/2.          ! NOT IN dB
         IF(I.LE.NSTART) GOTO 10
         IF(I.GT.NEND) GOTO 10
         TOTEN=TOTEN+DFEN(I)
           BLSUM(IBLOCK)=BLSUM(IBLOCK)+DFEN(I)
         II=(I-NSTART)/NPPBL*NPPBL
         IF(II.EQ.(I-NSTART)) THEN
            IBLOCK=IBLOCK+1
            BLSUM(IBLOCK)=0.0
         END IF
 10   CONTINUE
      DO 30 I=1,NBANDS
         BLPERC(I)=BLSUM(I)/TOTEN
 30   CONTINUE
      DO 40 I=1,NRATIO
         ITWICE=2*I
         RATIO(I)=BLPERC(NBRAT(ITWICE))/BLPERC(NBRAT(ITWICE-1))
 40   CONTINUE

C---------------------------
C     ACCUMULATED POWER CALCULATIONS
C---------------------------
      IPOW=1
      POW=ACCPOW(1)/100.*TOTEN
      FSUM=0.0
      DO 20 I=NSTART,NEND
         FSUM=FSUM+DFEN(I)
         IF(FSUM.GE.POW) THEN
            FACCPOW(IPOW)=FLOAT(I)/FLOAT(NEND-NSTART)
            IPOW=IPOW+1
            POW=ACCPOW(IPOW)/100.*TOTEN
            IF(IPOW.GT.NPOW) GOTO 21
         END IF
 20   CONTINUE
 21   RETURN
      END
```

D.2.1.3 Autocals Listing

```
C-------------------------------------------------------------
C     AUTOCORRELATION CALCULATIONS
C     CALC MEAN AND VARIANCE
C-------------------------------------------------------------
      SUBROUTINE AUTOCALS(AUTOC,NPAUTO,AVEAC,STDDEVAC)
      REAL AUTOC(60)

      SUM2X=0.0
      SUMX2=0.0
      AVEAC=0.0
```

```
      C1=1./FLOAT(NPAUTO)
      DO 10 I=1,NPAUTO
         AVEAC=AVEAC+AUTOC(I)*C1
         SUMX2=SUMX2+AUTOC(I)**2
         SUM2X=SUM2X+AUTOC(I)
 10   CONTINUE
      VARAC=(FLOAT(NPAUTO)*SUMX2-SUM2X**2)/
     1       (FLOAT(NPAUTO)*(FLOAT(NPAUTO)-1.))
      STDDEVAC=SQRT(VARAC)
      RETURN
      END
```

D.2.2 Tclass Listing

Tclass classified an unknown pattern by multiplying that pattern's feature vector by the weight vectors of the decision functions. Tclass was used to multiply the vectors only; Trun sent down the pattern vector and the relevant weight vectors.

```
C-------------------------------------------------------------
C
C     TCLASS CALLED BY TRUN TO CLASSIFY UNKNOWN PATTERNS
C     USING LINEAR DISCRIMINANT FUNCTIONS.
C     TCLASS RETURNS THE CLASS MEMBERSHIP OF THE PATTERNS
C     IN THE VARIABLE CC
C
C-------------------------------------------------------------
      SUBROUTINE TCLASS(CU,FEATNUM,W,M,CC)
      INTEGER FEATNUM,M,CC
      REAL W(20,200),D(20),CU(200),LARGE
      EXTERNAL WTMULT

      CALL WTMULT(W,CU,FEATNUM,D,M)

      LARGE=-9.E20
      DO 20 J=1,M
         LARGE=MAX(LARGE,D(J))
         IF(LARGE.EQ.D(J)) THEN
            CC=J
         END IF
 20   CONTINUE
      RETURN
      END
    SUBROUTINE TSNAP(IDAT,RPSDLA,ACA)
```

D.2.3 Tsnap Listing

The PSD and autocorrelation of the test recordings were calculated by Tsnap. A similar program, Snap, was used for the same purpose on the training patterns. This progam called ISML subroutines for the actual calculation of the PSD and autocorrelation.

```
C**************************************************
C
C     WRITTEN BY MIKE JONES , PRC KENTRON
C                APPLIED ACOUSTICS BRANCH
```

```
C                NASA LANGLEY RESEARCH CENTER
C
C*********************************************************
C
C            REFERENCES
C
C     1. RANDOM DATA: ANALYSIS AND MEASUREMENT PROCEDURLS by
C        Bendat and Piersol, Section 7.4, Pp.233-237.
C     2. INTRODUCTION TO TIME SERIES ANALYSIS by Jay C. Hardin,
C        NASA Ref Pub 1145, Section 6.5, Pp.55-56.
C     3. Same as (1), Table A.6, P.396.
C
C         PROGRAM DESCRIPTION
C
C     This program is designed to test data for
C               Autocorrelations
C           Power Spectral Densities
C
C     It breaks the pattern into NBL even blocks (and thus,
C     throws away extra points at end of data).
C     The data is then split into blocks of 4096 points each, and
C     AUTOCORRELATIONS and PSD's are computed on each block. Then, the
C     A-C's and PSD's are plotted (blocks and averages).
C
C     It is assumed that the T axis is a time axis, or one that acts
C     like a time axis, in that it is a constant increasing parameter.
C
C             VARIABLES
C
C     AC     AUTOCORRELATION OF EACH BLOCK
C     ACA     AVERAGE AUTOCORRELATION OF ALL BLOCKS
C     ACV     AUTOCOVARIANCE OF EACH BLOCK
C     BM     MEAN OF EACH BLOCK
C     BT     TOTAL TIME IN BLOCK
C     BV     VARIANCE OF EACH BLOCK
C     CALDB     CALIBRATION dB, LEVEL TO BE ADDED TO ALL PSD LEVELS
C     DF     DELTA FREQ FOR PSD PLOT
C     DTMS     DELTA TIME ASSOCIATED WITH X ARRAY, msec
C     FR     FREQ ARRAY FOR PSD PLOTS (IN kHz)
C     GS     GAUSSIAN PDF ARRAY BASED ON MEAN AND VARIANCE OF X
C     ICF     TDR CONFIGURATION CODE BYTE ARRAY
C     ICH     MEAS CHANNEL NUMBER
C     IDAT     INTEGER DATA ARRAY FROM TDR (NOT NORMALIZED)
C     IRCH     REF CHANNEL NUMBER
C     IRP     TDR RATE PROFILE BYTE ARRAY
C     ITIM     TIME OF DAY FOR TDR INITIATION
C     IWK     WORK ARRAY FOR FFTRC ROUTINE
C     NAC     # AUTOCORRELATIONS TO BE COMPUTED IN EACH BLOCK
C     NACM     MAX # AUTOCORRELATIONS TO BE COMPUTED IN EACH BLOCK
C     NBL     # BLOCKS OF DATA TO BE ANALYZED
C     NBN     # BINS TO USE IN PDF CALCULATION
C     NP     # POINTS IN X
C     NPM     MAX POSSIBLE # POINTS IN X
C     NPOPSD     # PNTS OUTPUT FROM PSD
C     NPPBL     # POINTS PER BLOCK
C     NPPSEG     # POINTS PER SEGMENT FROM TDR
C     NR     RUN NUMBER
C     NRM     # RUNS IN MEANS OF BLOCKS
C     NRV     # RUNS IN VARIANCES OF BLOCKS
C     PDF     PROB DENSITY FUNC OF DATA IN X (PLOTTED VS XD)
C     RPSD     PSD OF EACH BLOCK (IN VOLTS**2)
C     RPSDA     AVERAGE PSD OF ALL BLOCKS (IN VOLTS**2)
C     RPSDL     PSD OF EACH BLOCK (IN dB)
C     RPSDLA     AVERAGE PSD OF ALL BLOCKS (IN dB)
C     T     TIME ARRAY CORRESPONDING TO X (IN msec)
C     TM     MEAN OF TOTAL DATA SET
C     TV     VARIANCE OF TOTAL DATA SET
```

```
C     X    ARRAY OF INPUT AMPLITUDES
C     XAC    SCALE ARRAY FOR X AXIS OF AUTOCORRELATIONS
C     XMN    MIN VALUE IN X
C     XMX    MAX VALUE IN X
C     XD    X DIVISIONS ARRAY (DATA IN X IS PLACED INTO NBN BINS FOR
C         PLOTTING IN A PDF FORM)
C     YAC    SCALE ARRAY FOR Y AXIS OF AUTOCORRELATIONS
C
C
     PARAMETER (NACM=60, NPPBL=4096, NPOPSD=2049)
     PARAMETER (NPM=65536)
C-----------------------------------------------------------------------
C     THESE ARE CALCULATION PARAMETERS
C-----------------------------------------------------------------------
         PARAMETER (NAC=60,DTMS=.128041)
C-----------------------------------------------------------------------
     LOGICAL*1 TITAC(20), TITPSD(12), TITAC1(20)
C
C     ARRAYS FOR TDR SETUP
C
     INTEGER IDAT(NPM)
C
C     ARRAYS FOR RAW DATA, MEANS, VARIANCES
C
     DIMENSION BM(128), BV(128), T(NPOPSD), X(NPM)
C
C     ARRAYS FOR AUTOCORRELATION COMPUTATIONS
C
     DIMENSION AC(NACM), ACA(NACM), ACV(NACM), XAC(2), YAC(2)
C
C     ARRAYS FOR PSD COMPUTATIONS
C
     DIMENSION FR(NPOPSD), IWK(12), RPSD(NPOPSD), RPSDL(NPOPSD)
     DIMENSION RPSDA(NPOPSD), RPSDLA(NPOPSD)
     COMPLEX CPSD(NPOPSD)
C
     DATA IO, XMN, XMX/ 0, 1.E30, -1.E30/
C
C     CONSTANT ASSIGNMENTS
C
     DO I=1,NACM
      ACA(I)=0.
     END DO
     DO I=1,NPOPSD
      RPSDA(I)=0.
     END DO
     PI=4.*ATAN(1.)
     RDIV=204.9180328
C-----------------------------------------------------------------------
C     NEED TO KNOW TOTAL NUMBER OF POINTS AND DELTA T IN msec
C-----------------------------------------------------------------------
         NP=NPM

     NBL=NP/NPPBL              !# BLOCKS
     RNBL=NBL
     BT=NPPBL*DTMS*1.E-3         !TOT TIME PER BLOCK IN sec
     DF=1./BT              !DELTA FREQ IN HZ
     NP=NPPBL*NBL              !TOTAL # POINTS (EXTRA DATA NOT USED)
         DO I=1,NPM
       X(I)=FLOAT(IDAT(I))/RDIV      !CONVERT TO VOLTAGES
         END DO
C
C     COMPUTE MEAN VALUES OF TOTAL AND BLOCKS
C     COMPUTE VARIANCE VALUES OF TOTAL AND BLOCKS
C     COMPUTE AUTOCORRELATIONS
C
         TYPE 37
```

```
 37     FORMAT(/T10,'ENTER FIRST, LAST BLOCK OF RANGE TO BE'
     @      ' AVERAGED'$)
     READ(5,*,END=999,ERR=999)NSP1,NSP2
     IF(NSP1.EQ.0)NSP1=1
     IF(NSP2.EQ.0)NSP2=NBL
     DNBL=NSP2-NSP1+1
    XAC(1)=T(1)
    XAC(2)=T(NAC)
     NSP=-10
    ENCODE(20,22,TITAC1)NSP1,NSP2
 22    FORMAT('BLOCKS ',I3,' THRU ',I3,X)
    TYPE 35,NBL
 35    FORMAT(' BLOCK COUNTER OF ',I4/)
    DO J=1,NBL
     JJ=(J-1)/20*20
     IF(JJ.EQ.J-1)THEN
      TYPE 29,J
 29      FORMAT(X,I3,' ',$)
      ELSE
      TYPE 30,J
 30      FORMAT('+',I3,' ',$)
     END IF
     M=(J-1)*NPPBL
     IF(J.GE.NSP1 .AND. J.LE.NSP2)THEN
      CALL FTAUTO(X(M+1),NPPBL,NAC,0,5,BM(J),BV(J),
     @      ACV,AC,PACV,WK)
      DO K=1,NAC
       ACA(K)=ACA(K)+AC(K)/DNBL
      END DO
      ELSE
      NNAC=1
      CALL FTAUTO(X(M+1),NPPBL,NNAC,0,5,BM(J),BV(J),
     @      ACV,AC,PACV,WK)
     END IF
    END DO
C
C    COMPUTE PSD's
C
    TYPE 35,NBL
    DO J=NSP1,NSP2
     JJ=(J-1)/20*20
     IF(JJ.EQ.J-1)THEN
      TYPE 29,J
      ELSE
      TYPE 30,J
     END IF
     ENCODE(12,16,TITPSD)J
 16     FORMAT('BLOCK # ',I3,X)
     M=(J-1)*NPPBL
     CALL FFTRC(X(M+1),NPPBL,CPSD,IWK,WK)
     DO K=1,NPOPSD
      FR(K)=(K-1)*DF*1.E-3          !FREQ ARRAY IN kHz
      RPSD(K)=REAL(CPSD(K)*CONJG(CPSD(K)))*2.*PI/BT
      RPSD(K)=AMAX1(RPSD(K),1.E-3)
      RPSDA(K)=RPSDA(K)+RPSD(K)/DNBL
      RPSDL(K)=10.*ALOG10(RPSD(K))
     END DO
    END DO
C
C    COMPUTE AVE PSD
C
    DO K=1,NPOPSD
     RPSDLA(K)=10.*ALOG10(RPSDA(K))
    END DO
 999    CONTINUE
    RETURN
    END
```

D.3 CEO Listing

Ceo, named after chief executive officer, controlled the training and feature selection of the different classifiers. Ceo read in the training and evaluation sets as assembled by Fcals, then queried the type of operation; either train, select features, or classify unknown patterns. Ceo was used to classify unknown patterns in a preliminary testing phase and was not the program used for the testing presented in this study.

Ceo called several subroutines to complete the training, and these subroutines are included after the Ceo listing. Upon completion of training, Ceo stored the weight vectors and selected feature numbers in files specified by the user. Ceo had to be run for each classifer and each different number of features.

Feature selection was very time consuming, so this program was typically run in a batch operation, training several classifiers with all number of features in a single overnight run.

```
C-----------------------------------------------------------------
C
C       CEO - MAIN CONTROL PROGRAM -
C             CONTROLS READ IN, FEATURE SELECTION, AND CLASSIFICATION
C             OF PATTERNS BY CALLING THE SUBROUTINES SELECT AND
C             CLASSIFY
C
C     VARIABLES THAT MUST BE ENTERED -
C       FEATNUM - THE NUMBER OF FEATURES PER PATTERN
C       DESFEAT - THE DESIRED NUMBER OF BEST FEATURES TO SELECT
C       M - THE NUMBER OF CLASSES
C       ID - ID NUMBER FOR RUN
C
C
C-----------------------------------------------------------------
      INTEGER FEATNUM,DESFEAT,M,COUNT,EDUM,ECOUNT,BESTSET(30),
     1        PATNUM,CNUM,MODE,CC(200),SAT,AGAIN,
     2        ID,CHOIX
      REAL PATS(200,200),EVAL(200,200),CU(200),MISCLS,W(20,200),
     &     CRNTFT(200,200)
      LOGICAL ENUM,PRINT
      CHARACTER*24 TPATSFL,EPATSFL,WTMATFL,BSETFL,UNKFL
      EXTERNAL SELECT,BUILD

      WRITE(*,*)
      WRITE(*,*) '--- LINEAR DISCRIMINANT CLASSIFIER ---'
      WRITE(*,*) 'ENTER RUN ID NUMBER'
      READ(5,*) ID
 93   WRITE(*,*)
      WRITE(*,*) '                MAIN MENU       '
      WRITE(*,*)
      WRITE(*,*) '1. TRAIN SYSTEM'
```

```
      WRITE(*,*) '2. SELECT FEATURES AND TRAIN SYSTEM'
      WRITE(*,*) '3. CLASSIFY UNKNOWN PATTERNS'
      WRITE(*,*) '4. EXIT'
      WRITE(*,*)
 879  WRITE(*,*) 'ENTER YOUR CHOICE :'
      READ(5,*) CHOIX
      GOTO(1000,2000,3000,4000),CHOIX
         WRITE(*,*) 'PLEASE ENTER 1,2,3 OR 4'
         GOTO 879
C-------------------------------------------------------
C
C    TRAINING PHASE
C
C-------------------------------------------------------
 1000 WRITE(*,*)
      WRITE(*,*) 'TRAINING PHASE '

      REWIND 20
      WRITE(*,*) 'ENTER NAME OF FILE CONTAINING TRAINING PATTERNS'
      READ(5,1005) TPATSFL
 1005 FORMAT(A24)
      OPEN(UNIT=20,FILE=TPATSFL,STATUS='OLD')
      READ(20,*) NPATS,M,LN
      DO 1010 I=1,NPATS
         READ(20,*) (PATS(I,J),J=1,LN+1)
 1010 CONTINUE
      CLOSE(UNIT=20)
      LX=LN+1

      CALL BUILD(PATS,NPATS,LX,W,M,MISCLS,IT)

      WRITE(*,*)
      WRITE(*,*) 'PATTERNS TRAINED WITH',(100.*MISCLS),'   ERROR.'
      WRITE(*,*)
      WRITE(*,*) 'ENTER 1 TO SAVE RESULTS TO DISK, 0 OTHERWISE'
      READ(5,*) ITCH
      IF (ITCH.EQ.1) THEN
         REWIND 22
         WRITE(*,*) 'ENTER NAME OF FILE FOR WEIGHT MATRIX'
         READ(5,1005) WTMATFL
           OPEN(UNIT=22,STATUS='UNKNOWN')
         WRITE(22,*) ID,LN,M
         DO 1020 I=1,M
            WRITE(22,*) (W(I,J),J=1,LN+1)
 1020    CONTINUE
      END IF

      GOTO 93

C-------------------------------------------------------------------
C
C    FEATURE SELECTION AND CLASSIFIER TRAINING
C
C-------------------------------------------------------------------
 2000 WRITE(*,*)
      WRITE(*,*) 'FEATURE SELECTION AND CLASSIFIER TRAINING'
      WRITE(*,*)
      WRITE(*,*) 'ENTER THE DESIRED NUMBER OF FEATURES'
      READ(5,*) DESFEAT

      REWIND 20
      WRITE(*,*) 'ENTER NAME OF FILE CONTAINING TRAINING PATTERNS'
      READ(5,1005) TPATSFL
      OPEN(UNIT=20,FILE=TPATSFL,STATUS='OLD')
      READ(20,*) NTPATS,M,LN
      DO 2010 I=1,NTPATS
         READ(20,*) (PATS(I,J),J=1,LN+1)
```

```
2010 CONTINUE
     CLOSE(UNIT=20)

     WRITE(*,*)
     WRITE(*,*) 'ENTER 1 IF THERE ARE EVALUATION PATTERNS, 0 IF NONE'
     READ(5,*) NEPATS
     IF(NEPATS.NE.0) THEN
        ENUM=.TRUE.
        WRITE(*,*) 'ENTER NAME OF FILE CONTAINING EVALUATION PATS'
        READ(5,1005) EPATSFL
        OPEN(UNIT=21,FILE=EPATSFL,STATUS='OLD')
        READ(21,*) NEPATS,M,LN
        DO 2020 I=1,NEPATS
           READ(21,*) (EVAL(I,J),J=1,LN+1)
2020    CONTINUE
        CLOSE(UNIT=21)
     ELSE
        ENUM=.FALSE.
     END IF

     WRITE(*,*)
     WRITE(*,*) 'ENTER 1 TO SEE OUTPUT, 0 FOR NONE'
     READ(5,*) IRAN
     IF (IRAN.EQ.1) THEN
        PRINT=.TRUE.
     ELSE
        PRINT=.FALSE.
     END IF

     IF(NEPATS.EQ.0) THEN
        CALL SELECT(PATS,NTPATS,LN,DESFEAT,
    &            BESTSET,MISCLS,M,W,PRINT)
     ELSE
        CALL SELECT2(PATS,NTPATS,LN,DESFEAT,EVAL,NEPATS,
    &            BESTSET,MISCLS,M,W,PRINT)
     END IF
C-----------------------------------------------------------------
C    WRITE WEIGHT MATRIX AND PARAMETERS
C-----------------------------------------------------------------
     REWIND 22
     WRITE(*,*) 'ENTER NAME OF FILE FOR WEIGHT MATRIX'
     READ(5,1005) WTMATFL
     OPEN(UNIT=22,FILE=WTMATFL,STATUS='UNKNOWN')
     WRITE(22,*) ID,DESFEAT,M
     DO 2025 I=1,M
        WRITE(22,*) (W(I,J),J=1,DESFEAT+1)
2025 CONTINUE

     REWIND 23
     WRITE(*,*) 'ENTER NAME OF FILE FOR BESTSET FEATURES'
     READ(5,1005) BSETFL
     OPEN(UNIT=23,FILE=BSETFL,STATUS='UNKNOWN')
     WRITE(23,*) DESFEAT
     WRITE(23,*) (BESTSET(J),J=1,DESFEAT)
     CLOSE(UNIT=23)

     WRITE(*,*) 'RESULTS OF FEATURE SELECTION AND'
     WRITE(*,*) '    CLASSIFIER CONSTRUCTION     '
     WRITE(*,*)
     WRITE(UNIT=6,FMT=220) DESFEAT
220  FORMAT(1X,'TOP ',I3,' FEATURES:')
     WRITE(*,*) (BESTSET(I),I=1,DESFEAT)
     WRITE(UNIT=6,FMT=240) (100.*MISCLS)
240  FORMAT(1X,'ERROR RATE OF CLASSIFIER :',F6.2,' PERCENT.')

     GOTO 93
```

```
C-------------------------------------------------------------
C
C    CLASSIFY UNKNOWNS
C
C-------------------------------------------------------------
 3000 WRITE(*,*)
      WRITE(*,*) 'CLASSIFY UNKNOWNS'
      WRITE(*,*)

      REWIND 24
      WRITE(*,*) 'ENTER NAME OF FILE CONTAINING UNKNOWN PATTERNS'
      READ(5,1005) UNKFL
      OPEN(UNIT=24,FILE=UNKFL,STATUS='OLD')
      READ(24,*) IUNK,M,LN
      DO 3010 I=1,IUNK
         READ(24,*) (PATS(I,J),J=1,LN)
 3010 CONTINUE
      CLOSE(UNIT=24)


      WRITE(*,*)
      WRITE(*,*)
      WRITE(*,*) 'ARE THE UNKNOWNS IN BESTSET ORDER ? IF THE'
      WRITE(*,*) 'CLASSIFIER WAS TRAINED WITH A BESTSET OF FEATURES'
      WRITE(*,*) 'THE UNKNOWNS MUST BE IN A SIMILAR BESTSET TO OBTAIN'
      WRITE(*,*) 'MEANINGFUL RESULTS.'
      WRITE(*,*)
      WRITE(*,*) 'IF THE PATTERNS ARE IN BESTSET ORDER OR IF THERE IS'
      WRITE(*,*) 'NO BESTSET ORDER THEN ENTER 1, IF THERE IS A BESTSET'
      WRITE(*,*) 'ORDER AND THE PATTERNS ARE NOT IN THAT ORDER, ENTER 0
      READ(5,*) IBEST

      IF(IBEST.EQ.0) THEN
         REWIND 23
         WRITE(*,*) 'ENTER NAME OF FILE CONTAINING BESTSET FEATURES'
         READ(5,1005) BSETFL
         OPEN(UNIT=23,FILE=BSETFL,STATUS='OLD')
         READ(23,*) DESFEAT
         READ(23,*) (BESTSET(I),I=1,DESFEAT)
         CLOSE(UNIT=23)
         LN=DESFEAT
         IMD=0
         CALL ZUT(PATS,IUNK,CRNTFT,LN,BESTSET,IMD)

         DO 3030 I=1,IUNK
              DO 3030 J=1,LN
               PATS(I,J)=CRNTFT(I,J)
 3030    CONTINUE

      END IF

      REWIND 22
      WRITE(*,*) 'ENTER NAME OF FILE CONTAINING WEIGHT MATRIX'
      READ(5,1005) WTMATFL
      OPEN(UNIT=22,FILE=WTMATFL,STATUS='OLD')
      READ(22,*) IRID,IRLN,IRM
      DO 3020 I=1,M
         READ(22,*) (W(I,J),J=1,LN+1)
 3020 CONTINUE

      CALL CLASFY(PATS,IUNK,LN,W,M,CC)

      DO 290 I=1,IUNK
         WRITE(*,285) I,CC(I)
 285     FORMAT(1X,'PATTERN ',I3,' BELONGS TO CLASS ',I3)
 290  CONTINUE
      GOTO 93
```

```
4000 STOP
     END
```

D.3.1 Build Listing

Build was used by Select, which in turn was called by Ceo, for constructing a classifier from the features sent down. Build was called for every set of features, so in any given run it was called several hundred times to test the different sets of features.

```
C-------------------------------------------------------------------
C
C    SUBROUTINE BUILD - USING FEATURES SENT IN 'CRNTFT',
C    THIS BUILDS A CLASSIFIER USING MULTSUB AND RETURNS THE
C    ERROR RATE OF CLASSIFIER AS MISCLS
C
C    VARIABLES -
C       CRNTFT - MATRIX OF FEATURES
C       PATNUM - NUMBER OF PATTERNS IN CRNTFT
C    LX - AUGMENTED LENGTH OF CRNTFT
C    W - WEIGHT MATRIX AS CALCULATED BY BUILD
C    M - NUMBER OF CLASSES
C       MISCLS - ERROR RATE OF CLASSIFIER
C
C
C-------------------------------------------------------------------
      SUBROUTINE BUILD(CRNTFT,PATNUM,LX,W,M,MISCLS,IT)
      INTEGER PATNUM,LX,M,ITMAX,IT
      REAL CRNTFT(200,200),WT(20,200),MISCLS,EMIN,W(20,200)
      EXTERNAL MULTSUB,ETEST

      EMIN=2.0
      IT=0
      ITMAX=200

      CALL INIT(WT,M,LX)
      CALL INIT(W,M,LX)
 100  CALL MULTSUB(WT,CRNTFT,M,LX,PATNUM)
      CALL ETEST(CRNTFT,PATNUM,LX,WT,M,MISCLS)
      IT=IT+1
      EMIN=MIN(EMIN,MISCLS)
      IF(EMIN.EQ.MISCLS) THEN
         DO 10 I=1,M
            DO 10 J=1,LX
               W(I,J)=WT(I,J)
 10      CONTINUE
      END IF
      IF((EMIN.NE.0.0).AND.(IT.NE.ITMAX)) GOTO 100
      MISCLS=EMIN

      RETURN
      END
C-----------------------------------------------------
C
C    INIT IS USED TO ZERO WEIGHT MATRIX
C
C-----------------------------------------------------
      SUBROUTINE INIT(K,M,S)
      INTEGER M,S
      REAL K(20,200)
```

```
      DO 10 I=1,M
         DO 10 J=1,S
               K(I,J)=0.0
 10   CONTINUE
      RETURN
      END
```

D.3.2 Clasfy Listing

Clasfy is similar to Tclass except that Clasfy is used to classify a whole matrix of patterns, returning the membership of each in the array CC.

```
C-------------------------------------------------------------
C
C     CLASFY CALLED BY CEO TO CLASSIFY UNKNOWN PATTERNS
C     USING LINEAR DISCRIMINANT FUNCTIONS.
C     CLASFY RETURN THE CLASS MEMBERSHIP OF THE PATTERNS
C     IN THE ARRAY CC.
C
C-------------------------------------------------------------
      SUBROUTINE CLASFY(CRNTFT,PATNUM,FEATNUM,W,M,CC)
      INTEGER PATNUM,FEATNUM,M,CC(200)
      REAL CRNTFT(200,200),W(20,200),D(20),CU(200),LARGE
      EXTERNAL WTMULT

      DO 100 I=1,PATNUM
         DO 10 J=1,FEATNUM
            CU(J)=CRNTFT(I,J)
 10      CONTINUE

         CALL WTMULT(W,CU,FEATNUM,D,M)

         LARGE=-9.E20
         DO 20 J=1,M
         LARGE=MAX(LARGE,D(J))
            IF(LARGE.EQ.D(J)) THEN
               CC(I)=J
            END IF
 20      CONTINUE
 100  CONTINUE
      RETURN
      END
```

D.3.3 Multsub Listing

Multsub contains the perceptron algorithm. Multsub was used for training the decision functions by running through the set of training patterns once. The results of Multsub were then tested by Etest, to evaluate the success of the decision functions. Multsub was called by Build.

```
C-------------------------------------------------------------
C
C     MULTSUB - THIS SUBROUTINE GENERATES A WEIGHT MATRIX
C        TO CLASSIFY THE PATTERNS IN X USING THE ALGORITHM
C     FROM TOU AND GONZALEZ, PAGE 181-.
C
```

```
C    VARIABLES -
C    M - NUMBER OF CLASSES
C    LX - LENGTH OF PATTERN VECTOR, CLASS + AUGMENTED
C    N - NUMBER OF TRAINING PATTERNS
C    W - WEIGHT VECTOR, CONTAINS DECISION FUNCTION
C    X - TRAINING PATTERNS, LENGTH M*L; FIRST ENTRY CONTAINS
C        CLASS OF PATTERN VECTOR
C    CU - CURRENT PATTERN, LENGTH F
C    CL - CLASS OF CURRENT PATTERN
C    D - VALUES OF DECISION FUNCTION, LENGTH M
C    FLAG - INCORRECT DECISION FLAG
C    CLK - CLASS CHECK
C    CO - NUMBER CORRECTLY CLASSIFIED
C
C  ***  MAXIMUM ALLOWABLE NUMBER OF CLASSES AND FEATURES :   ***
C  ***               20 CLASSES, 199 FEATURES                 ***
C
C-----------------------------------------------------------------
      SUBROUTINE MULTSUB(W,X,M,LX,N)
      INTEGER M,N,LX,CL,CLK
      REAL W(20,200),X(200,200),LARGE,D(20),CU(200)
      LOGICAL FLAG
      EXTERNAL WTMULT

 1    DO 40 I=1,N
         DO 10 J=1,(LX-1)
            CU(J)=X(I,J+1)
 10      CONTINUE
         CL=INT(X(I,1))
         CU(LX)=1.0

         CALL WTMULT(W,CU,LX,D,M)

C-----------------------------------------------------------------
C    CHECK VALUES OF DECISION FUNCTION
C-----------------------------------------------------------------
         FLAG=.FALSE.
         DO 30 J=1,M
            IF(J.EQ.CL) GOTO 30
            IF(D(J).GE.D(CL)) THEN
               FLAG=.TRUE.
               SIGN=-1.0

               CALL ADJUST(W,CU,LX,SIGN,J,M)

            END IF
 30      CONTINUE
         IF (FLAG) THEN
            SIGN=1.0

            CALL ADJUST(W,CU,LX,SIGN,CL,M)

         END IF
 40   CONTINUE

      RETURN
      END

C-----------------------------------------------------------------
C    ADJUST MAKES CORRECTION TO WEIGHT VECTOR DEPENDING
C     ON WHETHER OR NOT CORRECT DECISION WAS MADE
C-----------------------------------------------------------------
      SUBROUTINE ADJUST(W,CU,LX,SIGN,CL,M)
      INTEGER LX,CL,M
      REAL W(20,200),CU(200),SIGN

      DO 10 I=1,LX
```

```
         W(CL,I)=W(CL,I)+SIGN*CU(I)
 10    CONTINUE

       RETURN
       END
```

D.3.4 Etest Listing

Etest was used to evaluate the error rate of a given classifier using an evaluation set of patterns. Etest was called by the feature selction routine, Select, in order to test the performance of each set of patterns, when used to construct decision functions in the subroutine Build.

```
C-----------------------------------------------------------------
C
C     EVALUATES THE CLASSIFIER FROM BUILD USING AN
C     EVALUATION SET OF PATTERNS. THE ERROR RATE IS IN MISCLS
C
C     VARIABLES -
C     EVAL - MATRIX OF EVALUATION PATTERNS
C     N - NUMBER OF PATTERNS IN EVALUATION SET
C     L - NUMBER OF FEATURES+1 PER PATTERN
C     W - WEIGTH MATRIX FROM BUILD
C     M - NUMBER OF CLASSES
C     MISCLS - ERROR RATE OF CLASSIFIER
C
C  *** MAX NUMBER OF CLASSES = 20, MAX NUMBER OF ***
C  ***           FEATURES EQUALS 29                ***
C  ***    MAX NUMBER OF TRAINING PATTERNS =100    ***
C
C-----------------------------------------------------------------
       SUBROUTINE ETEST(EVAL,N,L,W,M,MISCLS)
       INTEGER N,L,M,CL,CO,CLK
       REAL EVAL(200,200),W(20,200),CU(200),D(20),LARGE,RT,MISCLS
       EXTERNAL WTMULT

       NCO=0
       CU(L)=1.0

       DO 100 I=1,N
          DO 10 J=1,L-1
             CU(J)=EVAL(I,J+1)
 10       CONTINUE
          CL=INT(EVAL(I,1))

          CALL WTMULT(W,CU,L,D,M)

          LARGE=-9.E20
          DO 50 J=1,M
             LARGE=MAX(LARGE,D(J))
             IF(LARGE.EQ.D(J)) THEN
                CLK=J
             END IF
 50       CONTINUE
          IF(CLK.NE.CL) THEN
             NCO=NCO+1
          END IF
 100   CONTINUE
       MISCLS=(FLOAT(NCO))/(FLOAT(N))

       RETURN
```

```
      END
```

D.3.5 Select Listing

Select was called by Ceo in order to select a best set of features with a set of training and evaluation patterns read in by Ceo. The nuber of features to be selected was sent down by Ceo. Since the selection of features required the construction of a set of decision functions using those features, Select also returned the coefficients of the decision functions along with the numbers of the best set of features.

```
C-----------------------------------------------------------------
C     SELECT2 - FEATURE SELECTION PROGRAM USING GPRI ALGORITHM
C               FROM VPI&SU THESIS BY KLASSEN
C     THIS ALGORITHM SELECTS THE (DESFEAT) BEST FEATURES FROM
C     A CANDIDATE LIST
C
C     VARIABLES -
C        FEATNUM - NUMBER OF FEATURES IN CANDIDATE LIST
C        PATNUM - NUMBER OF PATTERNS IN TRAINING SET
C        DESFEAT - THE DESIRED NUMBER OF FEATURES
C        LENFRANK - THE LENGTH OF THE FRANK ARRAY
C        ELIM - NUMBER OF TIMES A FEATURE IS DISCARDED BEFORE IT IS
C               ELIMINATED
C        FRANK - CONTAINS THE RANKINGS OF THE CANDIDATE FEATURES,
C                ACCORDING TO THEIR F RATIO. ALSO USED AS
C                BOOKEEPING ARRAY - THE TOP FEATURE IN FRANK IS THE
C                NEXT FEATURE TO BE TESTED
C        DELCOUNT - CONTAINS THE NUMBER OF TIMES FEATURE HAS BEEN
C                   DISCARDED
C        BESTSET - CONTAINS THE CURRENT (DESFEAT) BEST FEATURES
C        ETHRESH - A FEATURE MUST IMPROVE CLASSIFICATION BY THIS
C                  MUCH TO BE ACCEPTED
C
C *** ARRAY DIMENSIONS MUST BE INCREASED IF NUMBER OF FEATURES ***
C *** IS GREATER THAN 199 OR NUMBER OF TRAINING        ***
C *** PATTERNS IS GREATER THAN 100                              ***
C-----------------------------------------------------------------
      SUBROUTINE SELECT2(X,PATNUM,FEATNUM,DESFEAT,EVAL,ENUM,BESTSET,
     &                   MISCLS,M,W,PRINT)
      INTEGER FEATNUM,PATNUM,DESFEAT,LENFRANK,LEVEL,ELIM,TEST,HOLD,
     &        DISCARD,L,TEMP,OUT,ENUM,M,LX,MODE,LCRNT
      INTEGER FRANK(200),DELCOUNT(200),BESTSET(30),IT,NHOLDBEST(30)
      REAL X(200,200),EVAL(200,200),
     &     CRNTFT(200,200),ERROR(200),W(20,200)
      REAL ETHRESH,BESTERROR,P2ERR(200),ERRMIN,MISCLS
      LOGICAL PRINT,GOOD
      EXTERNAL ZUT,BUILD,ETEST

      LX=FEATNUM+1
      ELIM=3
      ETHRESH=.005
      MODE=1
      NAMY=0

      DO 5 I=1,FEATNUM
         DELCOUNT(I)=0
 5    CONTINUE
```

```
      LENFRANK=FEATNUM
      LEVEL=1

C     CALL ORDER(X,PATNUM,LX,FRANK,M)
      DO 83 I=1,LX-1
         FRANK(I)=I
 83   CONTINUE


      IF(PRINT) THEN
         WRITE(6,7) DESFEAT
 7       FORMAT(1X,I3,' FEATURES BEING TESTED AND PERCENT ERROR')
      END IF
C-----------------------------------------------------------------------
C     INITIALIZE BESTSET WITH TOP FEATURE
C-----------------------------------------------------------------------
      BESTSET(1)=FRANK(1)
      LENFRANK=LENFRANK-1
      CALL SHIFT2(FRANK,LENFRANK)

      CALL ZUT(X,PATNUM,CRNTFT,LEVEL,BESTSET,MODE)
      CALL BUILD(CRNTFT,PATNUM,2,W,M,MISCLS,IT)
      IT=0
      IF (ENUM.NE.0) THEN
         CALL ZUT(EVAL,ENUM,CRNTFT,LEVEL,BESTSET,MODE)
         CALL ETEST(CRNTFT,ENUM,2,W,M,MISCLS)
      END IF
      ERROR(LEVEL)=MISCLS
C-----------------------------------------------------------------------
C        PHASE I
C-----------------------------------------------------------------------
 10   IF(LENFRANK.EQ.0) GOTO 99
      TEST=FRANK(1)
      BESTSET(LEVEL+1)=TEST
      LEVEL=LEVEL+1

      CALL ZUT(X,PATNUM,CRNTFT,LEVEL,BESTSET,MODE)
      CALL BUILD(CRNTFT,PATNUM,(LEVEL+1),W,M,MISCLS,IT)
      IT=0
      IF (ENUM.NE.0) THEN
         CALL ZUT(EVAL,ENUM,CRNTFT,LEVEL,BESTSET,MODE)
         CALL ETEST(CRNTFT,ENUM,(LEVEL+1),W,M,MISCLS)
      END IF
      ERROR(LEVEL)=MISCLS
      CALL SHIFT2(FRANK,LENFRANK)
      IF(ERROR(LEVEL).LE.ERROR(LEVEL-1)) THEN
C---FEATURE(TEST) SUCCESSFUL
         LENFRANK=LENFRANK-1
         IF(LEVEL.EQ.DESFEAT) GOTO 99
      ELSE
C---FEATURE NOT SUCCESSFUL
         LEVEL=LEVEL-1
         DELCOUNT(TEST)=DELCOUNT(TEST)+1
         IF(DELCOUNT(TEST).GE.3) THEN
            LENFRANK=LENFRANK-1
         ELSE
            FRANK(LENFRANK)=TEST
         END IF
      END IF
      GOTO 10
C-----------------------------------------------------------------------
C   BEGINNING OF PHASE II
C-----------------------------------------------------------------------
 99   BESTERROR=ERROR(DESFEAT)
      IF (PRINT) THEN
         IMIS=INT(100.*MISCLS)
         WRITE(6,8) (BESTSET(I),I=1,DESFEAT),IMIS
```

```
 8       FORMAT(20I5)
      END IF
 100  IF(LENFRANK.EQ.0) GOTO 500
      NAMY=NAMY+1
      IF(PRINT) THEN
         WRITE(*,*) 'ITERATIONS =',NAMY
      END IF
      IF((NAMY.EQ.(2*DESFEAT)).OR.(NAMY.EQ.(4*DESFEAT)).OR.
     1     (NAMY.EQ.(6*DESFEAT))) THEN
           DO 898 I=1,DESFEAT
              NHOLDBEST(I)=BESTSET(I)
 898       CONTINUE
           DO 899 I=0,DESFEAT-1
              BESTSET(I+1)=NHOLDBEST(DESFEAT-I)
 899       CONTINUE
      END IF
C---PICK OUT NEXT FEATURE TO BE TESTED
      TEST=FRANK(1)
      ERRMIN=1.01
      DO 20 I=1,DESFEAT
         HOLD=BESTSET(I)
         BESTSET(I)=TEST

         CALL ZUT(X,PATNUM,CRNTFT,DESFEAT,BESTSET,MODE)
         CALL BUILD(CRNTFT,PATNUM,(DESFEAT+1),W,M,MISCLS,IT)
      IT=0
      IF (ENUM.NE.0) THEN
         CALL ZUT(EVAL,ENUM,CRNTFT,DESFEAT,BESTSET,MODE)
         CALL ETEST(CRNTFT,ENUM,(DESFEAT+1),W,M,MISCLS)
      END IF

      IF(PRINT) THEN
         IMIS=INT(100.*MISCLS)
         WRITE(6,9) (BESTSET(J),J=1,DESFEAT),IMIS
 9       FORMAT(20I5)
      END IF
         P2ERR(I)=MISCLS
         BESTSET(I)=HOLD
         ERRMIN=MIN(ERRMIN,P2ERR(I))
         IF(ERRMIN.EQ.P2ERR(I)) THEN
            GOOD=.TRUE.
         ELSE
            GOOD=.FALSE.
         END IF
         IF(GOOD) THEN
C---LEAVING BESTSET(I) GIVES BEST RESULTS IN COMBO WITH TEST ----------
            OUT=I
         END IF
 20   CONTINUE

      IF(ERRMIN.GT.BESTERROR) THEN
C---TEST FEATURE DID NOT IMPROVE PERFORMANCE
         DISCARD=TEST
      ELSE
C---TEST FEATURE DID IMPROVE PERFORMANCE
         DISCARD=BESTSET(OUT)
         BESTSET(OUT)=TEST
      END IF

      CALL SHIFT2(FRANK,LENFRANK)
      DELCOUNT(DISCARD)=DELCOUNT(DISCARD)+1
      IF (DELCOUNT(DISCARD).LT.3) THEN
         FRANK(LENFRANK)=DISCARD
      ELSE
         LENFRANK=LENFRANK-1
      END IF
      BESTERROR=MIN(ERRMIN,BESTERROR)
```

```
      GOTO 100

 500  CALL ZUT(X,PATNUM,CRNTFT,DESFEAT,BESTSET,MODE)
      CALL BUILD(CRNTFT,PATNUM,(DESFEAT+1),W,M,MISCLS,IT)
      RETURN
      END

      SUBROUTINE SHIFT2(F,L)
C---USED TO SHIFT MEMBERS OF FRANK SO BEST IS IN TOP POSITION
      INTEGER L
      INTEGER F(200)
      DO 10 I=1,L
         F(I)=F(I+1)
 10   CONTINUE
      RETURN
      END
```

D.3.6 Wtmult Listing

Wtmult was one of the most widely utilized subroutines with the simplest purpose; multiply a pattern vector and a weight vector and return the result. This subroutine was used by Clasfy to classify patterns, and Multsub for training the patterns.

```
C-------------------------------------------------------------
C
C     WTMULT MULTIPLIES THE WEIGHT MATRIX BY THE PATTERN
C     CU, GIVING VALUES FOR EACH OF THE M DECISION
C     FUNCTIONS
C
C-------------------------------------------------------------
      SUBROUTINE WTMULT(W,CU,F,D,M)
      INTEGER F,M
      REAL W(20,200),CU(200),D(20)
      DO 10 I=1,M
         D(I)=0.0
         DO 10 J=1,F
              D(I)=D(I)+W(I,J)*CU(J)
 10   CONTINUE
      RETURN
      END
```

D.3.7 Zut Listing

Zut was used for transferring feature values into feature sets. A best set of features was selected from a candidate list, but every pattern's feature vector contained every feature in the candidate list. In order to evaluate the performance of only a small subset of these features, Zut was used to transfer the feature values of interest into a new pattern matrix.

```
C-------------------------------------------------------------
C     ZUT TAKES PATTERNS AND DESIRED FEATURES FROM THE
C     MATRIX OF PATTERNS AND PLACES THEM IN THE MATRIX CRNTFT
C
C     VARIABLES -
```

```
C     X - MATRIX CONTAINING TRAINING PATTERNS
C     PATNUM - NUMBER OF PATTERNS IN X
C        CRNTFT - NEW MATRIX OF PATTERNS AND FEATURES
C      LEVEL - NUMBER OF FEATURES TO BE MOVED
C     BESTSET - FEATURE NUMBERS TO BE MOVED
C
C
C-----------------------------------------------------------------
      SUBROUTINE ZUT(X,PATNUM,CRNTFT,LEVEL,BESTSET,MODE)
      INTEGER PATNUM,LEVEL,MODE,BESTSET(30)
      REAL X(200,200),CRNTFT(200,200)

      DO 10 I=1,PATNUM
         CRNTFT(I,1)=X(I,1)
         DO 10 J=1,LEVEL
            CRNTFT(I,(J+MODE))=X(I,(BESTSET(J)+MODE))
 10   CONTINUE

      RETURN
      END
```

The vita has been removed from the scanned document