

NEURAL NETWORKS IN FAULT DETECTION: A Case Study

1

D.R. Hush, C.T. Abdallah, G.L. Heileman, and D. Docampo
EECE Department,
University of New Mexico,
Albuquerque, NM 87131, USA.

Abstract

In this paper we study the applications of neural nets in the area of fault detection. In particular, neural networks are used for fault detection in real vibrational data. The study is one of the first to include a large set of real vibrational data and to illustrate the potential as well as to the limitations of neural networks for fault detection.

1. Introduction

There has been considerable work in the areas of fault detection and isolation which were reviewed in [4], [15], [8]. We first comment on standard designs to detect faults using *hardware redundancy*. This approach relies on duplicating the functionality of some critical components in the system, and on using a majority voting logic to decide on the presence, location, and type of faults. Such an approach is of course costly and requires the anticipation of faults types and locations.

On the other hand, the *analytical redundancy* (also termed functional redundancy) approach exploits the inherent redundancy in the relationships between the system's inputs and outputs. The approach is appealing in terms of hardware cost but can also suffer from poor models and may require sophisticated signal processing techniques in order to extract useful information out of the data.

There are basically two generic ways to approach the analytical fault detection problem: The model-based approach and the data-based approach.

In the model-based approach, the engineer has access to a model of the system whose behavior is being monitored. The model could be analytical, or knowledge-based. Most applications of this approach

have dealt with linear systems, since they can be easily described and studied.

In the data-based approach one bypasses the step of obtaining a mathematical model and deals directly with the data. This is more appealing when the process being monitored is not known to be linear or when it is too complicated to be extracted from the data. It is this approach which we will concentrate on in this paper in order to evaluate the potential of neural networks as fault detectors. We can divide the data-based approach into a *time-domain techniques* and *frequency-domain techniques*. Note that either technique is actually a front-end of the final fault detection systems. In fact, these techniques provide different indicators which are then combined in a classifier to provide a test for the existence and type of faults. This paper discusses the potential, as well as the limitations, of neural nets usage in fault detection and possible accommodation in vibrational systems.

In order to address the issues discussed above, this report is organized as follows. Section 2 provides a study of NN in fault detection. Our approach to the fault detection and isolation problem is presented and our results obtained using real data are given in section 3. We have also investigated a trending approach using Fuzzy ART, the results of which are presented in section 4. Finally, our conclusions and recommendations are provided in section 5.

2. Applications of Neural Networks in Fault Detection

It is conceivable that a neural net can be used as a monitoring device, in order to detect major changes in the operation of the system. Specifically, one approach may be that the neural net is trained on a well-behaving system, and then operated with no more training in parallel to the actual system. The neural net output will then be compared to that of the physical system, and any anomalies in the output of our system will be detected. This approach,

¹The research of D. Hush, C.T. Abdallah, and G. Heileman was supported by a Grant from Chadwick-Helmuth under Contract W-300445. The authors are grateful to JP Cain of Chadwick-Helmuth for his help in collecting the data and his guidance. The authors also acknowledge fruitful discussions and help from Mr. Jose Luis Alba, of the University of Vigo, Spain and the generous support of ISTEAC.

labeled “trending” was not originally used in our research. However, we have included a section to discuss some preliminary results using Fuzzy ARTMAP and the trending approach at the end of the paper.

In some cases, the neural net may also be used to accommodate the change of behavior as part of hierarchical control system [11]. In others, it is used simply as a fault detection device where the clustering capabilities of networks such as ART or CMAC are called upon [7]. As always, it is a bad idea to blindly apply the measured data to the input of a NN. First and foremost, the given data may be high dimensional which may greatly increase the number of weights in the NN and slow down its training algorithm. Instead, it is advisable to pre-process the data in order to obtain some important features, thus reducing the dimensionality of the training data, and with it the number of required weights. Such pre-processing may be as simple as obtaining the FFT coefficients [10, 14] for a stationary time series, or more advanced approaches for nonstationary time series or for data which may not be separable based on the spectral content alone.

The general idea behind using a NN for fault detection can be summarized in the following steps:

1. Use a signal processing techniques to obtain a figure of merit f (Spectrum, Cepstrum, k factor, etc) for the different time signals.
2. If the figure of merit is high dimensional, use a feature extraction algorithm to reduce its dimensionality while keeping most of its information content. The resulting signal is f_e .
3. Train the neural network on f_e either in supervised or unsupervised mode as discussed below.

Neural networks can be trained in one of two modes: Supervised learning when input/output data is available, or when both “good” and “bad” data is available, and un-supervised learning when only output data is available, or when has no knowledge of the quality of his data.

When one has access to input/output data and would like to obtain a functional mapping from the input to the output, a supervised learning scheme is used. Such an approach may also be used if one has labeled data (“good”, “bad”) in order to separate the good from the bad. In a fault detection application, where the designer knows that some data corresponds to the presence and the absence of particular faults, this approach becomes obvious. The difficulty with this approach is that some data labeled “normal” may in fact be bad. Also, in order to

work properly, a representative sample of all faults should be included in the training set.

One needs to use unsupervised learning when the collected data is not labeled. A neural network is then operated as a pattern classifier, trying to discover similarities between some data and anomalies amongst others. Such an approach was used in [2] where an ART NN was used to cluster “normal” behavior of the process. After training has stopped, the NN weights are fixed and when a substantially different time series data is presented to the network, it would be detected as sufficiently outside any of the clusters formed so far.

3. Detecting Faulty Bearings: A Feasibility Study

The purpose of this study is to explore automated methods for detecting faults in the viscous damper bearing of a helicopter drive shaft. This portion of the paper describes the design of a system that accomplishes this task using *neural networks*.

3.1. Data Description

The data used in this portion of the study are *time series data* measured by an accelerometer located on the outer bracket housing of the helicopter shaft (measured with respect to the shaft 1P). The time series were sampled at a rate of 44,100 sps. The bandwidth of the anti-aliasing filter was approximately 20 KHz. Spectrograms of the data (shown below) suggest that the signals are *stationary*, and that they were sufficiently oversampled (i.e. there was virtually no power above 10 KHz).

A total of 21 runs were made using 17 different bearings. The duration of the runs varied from 4 to 7 seconds. Runs 1 and 11 were discarded because they represent “start-up” runs from the two different data gathering trips, and were both suspect in one way or another.

Thus, the input data for this portion of the study consists of time series data collected from the outer bracket accelerometer for Runs 2–10 and 12–21. All experiments described in this section used approximately 4 seconds of data from each run. Approximately 2.8 seconds (70%) were used for training and 1.2 seconds (30%) for testing. The stationary nature of the signals, coupled with the abundance of data samples, made it unnecessary to perform full cross-validation in our experiments. As we shall see, all experiments produced extremely close agreement between training and test performance with a single split of the data.

The second level of categorization placed the runs into *two* groups: **Good** and **Bad**. Runs that

we present a preliminary study of this problem using an ART neural network.

Run Numbers	Level I Grouping	Level II Grouping
7,12,14	New	Good
8,13,15,16	Used	Good
2,9,17,19	Ball Spall	Bad
10,18,20	Inner Race	Bad
5	Outer Race	Bad
6,21	Other	Bad

Table 1: Summary of Categorizations for Runs 2–10 and 12–21.

were categorized as *New* or *Used* where considered **Good** and the rest were considered **Bad**. Table 1 summarizes the two levels of groupings.

3.2. System Design

The ultimate goal here is to design a system that will correctly categorize time series data from a helicopter drive shaft into one or both of the groupings shown in Figure 1. Our focus is on the Level II grouping (i.e. *good* versus *bad*). The system designed here is *data-driven*, i.e. data with known categorizations are used to design various components of the system. Such a system can only be guaranteed to work if the data used in the design is truly representative of data that will be presented to the system in the future. It is unclear whether this is actually true for this particular problem. It has been our experience (as well as others [?]) that there is wide variability in the signature produced by both good and faulty bearings and that the overlap in these signatures is quite large. In other words, the signature of a good bearing is often nearly identical to that of a faulty bearing (and vice versa). This problem stems from the fact that there are *numerous factors* that contribute to the variability among bearing signatures, and the *quality* of the bearing is *only one of these factors* (and not always the dominate one). Thus, while the system designed below may achieve good generalization performance on the data gathered for this study, its performance on future data is difficult to predict.

An alternative approach would be to treat each bearing as a separate problem. A fault could be detected by monitoring the signature of the bearing over the course of its lifetime and raising a flag when the bearing’s signature begins to deviate appreciably from its starting value. This approach has been used with a great deal of success on problems very similar to the one under consideration here [?]. Although a different type of data would be required to investigate this approach (i.e. we would need data collected over the lifetime of several different bearings),

The basic approach followed here is outlined in Figure ?? This is a traditional pattern recognition system consisting of three major components: preprocessing, feature extraction and classification. The purpose of the preprocessing stage is generally to compensate for known distortions introduced by the sensor and/or the environment. This stage often involves operations such as scaling, resampling, and/or filtering. The only preprocessing performed on the bearing data was to normalize each of the runs so that it had zero mean and unit standard deviation.

The second stage is generally the most difficult to optimize. The purpose of feature extraction is “to extract features from the preprocessed data which provide the *greatest discrimination* between pattern classes”. This is often difficult because the “best features” are usually not known ahead of time. Perfecting this stage generally involves an in-depth knowledge of the problem at hand, good intuition, and a fair amount of trial and error. In the bearing classification problem we don’t know what features will work well ahead of time, so our approach has been to use features that are either commonly used for other types of time series data and/or have been found to work well in other studies involving the detection of faulty bearings from accelerometer data. To this end we settled on the following three feature sets:

1. *Spectrograms*: A time-varying estimate of the magnitude of the Fourier Transform of the data.
2. *Linear Prediction Coefficients (LPC)*: Coefficients of an optimal M^{th} order (FIR) linear predictor for the data.
3. *Cepstrum*: The inverse Fourier transform of the logarithm of the magnitude of the Fourier transform of the data.

In addition to their ability to carry discriminatory information, a good feature set should also have the following properties:

Invariance: The features should be *invariant* to superfluous variations in the data, i.e. variations in the data that provide no discriminatory information.

Dimensionality Reduction: The training process and generalization performance of the classifier (the last stage) suffer from the curse of dimensionality. It is therefore important to reduce the dimensionality (i.e. the number of features)

as much as possible at the feature extraction stage.

Simplified Representation: Ideally, the features should take on a representation that permits optimal discrimination with the simplest possible classifier. For example, it is common to strive for representations in which the two classes are *linearly separable*.

The final stage in Figure ?? is the classifier. We investigated a wide variety of classifiers in this study, with a focus on the following *neural network* and *fuzzy* classifiers:

1. Multilayer Perceptrons (MLPs)
2. Radial Basis Functions (RBFs)
3. Fuzzy ARTMAP

In addition to these we investigated the traditional *linear*, *quadratic* and *nearest-neighbor* classifiers.

All systems designed in this report were comprised of the five stages shown in Figure ?. The normalization stage (zero mean and unit standard deviation) has already been described. The details of the remaining four stages are described in the sections that follow.

3.3. Feature Computation

The computational aspects of the three feature sets, Spectrograms, LPC coefficients and Cepstral coefficients, are described below.

Spectrograms: The spectrograms were computed as follows. The first 4 seconds (176,000 samples) of each run were partitioned into 500 consecutive nonoverlapping time segments and a spectral estimate was formed from the 352 samples in each segment. Spectral estimates were obtained using Welch's method (averaging modified periodograms). Each modified periodogram was formed by applying a Hamming window of length 64 (256) to the time samples, computing the FFT, and then taking the squared magnitude for each resulting frequency bin. A total of 6 (2) periodograms were averaged to form the spectral estimate for each 352 sample segment. In the end, each estimate contained 33 (129) frequencies at equally spaced intervals from d.c. to Nyquist (i.e. 0 to 22 KHz).

LPC Coefficients: Linear Prediction Coefficients (LPCs) were computed as follows. Approximately 4 seconds (179,200 samples) of each run

were partitioned into 700 consecutive (overlapping) time segments of length 1024. Consecutive time segments were overlapped by 768 samples (i.e. the segment window was shifted by 256 samples at each step). From each 1024-sample segment 16 Linear Prediction Coefficients were computed using the Levinson-Durbin algorithm [12].

Cepstrum: Cepstral coefficients were computed from the LPC coefficients using the method described in [?] (problem 12.32, p. 833). A total of 16 cepstral coefficients are computed for each time segment.

Plots of the spectrograms for each of the 19 runs used in this study are shown in Figures ??-?. Several observations can be made from these plots.

1. Except for Run 2, there is no noticeable power in the upper half of the frequency range (i.e. from 11 to 22 KHz).
2. In many cases there are strong similarities between the runs for good and bad bearings. For example, compare Runs 8 and 10, or Runs 13 and 19, or Runs 12 and 21.
3. Two separate runs of the same bearing often have noticeable differences. For example compare Runs 7 and 12.
4. Although there are notable exceptions, the following trend seems apparent. Good bearings (both *new* and *used*) tend to have most of their spectral power concentrated in one low frequency peak. As the bearing becomes faulty the power tends to spread into higher frequencies, often introducing a second and/or third peak. For example, see the sequence of Runs 13, 16, 18 and 20. Runs 13 and 16 are the same (normal) bearing (16 a take-apart version of 13) and Runs 18 and 20 have increasing amounts of damage.

3.4. Feature Selection

The purpose of this section is to determine which of the *individual* features in the three feature sets are useful for discrimination. Those that are not can be discarded. There are numerous ways of approaching this problem [?, 5]. The method used here forms an estimate of the Bayes Classification Error (i.e. the minimum attainable classification error) for each individual feature, and discards features with a high (close to 50 %) error. That is, each feature is used *by itself* as a discriminator, and an estimate of its

best performance is obtained. Those features with high individual error rates can safely be discarded since they will never contribute significantly to the discrimination problem.

It should be noted that this method filters *individual* features only. It makes no attempt to reject features which carry redundant discriminatory information (e.g. features that are highly correlated).

The Bayes Classification Error for each feature is estimated using the k -nearest neighbor (k -NN) classifier (with *reject* in the case of a tie) and the *leave-one-out* method of cross-validation [5]. Separate estimates of classification error are obtained for odd and even values of k . In theory the classification error for odd and even values of k provide an upper and lower bound (respectively) on the actual Bayes error. Also, as k grows large (and the number of samples approaches ∞), these bounds become tighter, so that in the limit, the even and odd classification errors merge at the true Bayes error [5]. To “estimate” the Bayes error, the classification error of the k -NN classifier is estimated (using leave-one-out) for values of k ranging from 1 to 20, and an estimate of the convergent point for the even and odd curves is obtained.

Plots of the Bayes error estimates for the Spectral features are shown in Figure ???. The two plots correspond to estimates for the *low* and *high* frequency resolution features. Both plots have the same general shape. The lower resolution features, however, achieve uniformly better classification scores. This, coupled with the fact that initial attempts to classify the data using the high resolution features provided no improvement over the low resolution features (in fact the performance was slightly worse), prompted us to omit the high resolution features from further study.

For the low resolution spectral features, the plot in Figure ??? clearly shows the highest degree of discrimination in the frequency range 5–8 KHz, the same region where the second peak often appears for a bad bearing. In addition, the error estimate for most of the remaining frequencies is below 40%, suggesting that it may be unwise to simply discard them. On the other hand, a visual inspection of the spectrograms in Figures ???–???, coupled with the results of previous studies and a scientific intuition about this problem, suggest that the upper half of the spectrum does not carry information useful for discriminating between good and bad bearings. The discrimination in this frequency range suggested by the plots in Figure ??? is most likely due to other types of variations in the data (not variations in the quality of the bearing). For this reason we chose to discard the upper

17 (of 33) frequencies. The lower 16 are all retained for further processing.

The Bayes error estimates for the LPC and Cepstral features are shown in Figure ???. Both contain 3 or 4 coefficients with relatively high classification error, but because there are so few, and the dimensionality of these features is low to begin with (only 16), we decided to retain all of them.

3.5. Dimensionality Reduction

The previous section explored methods of discarding individual features. In this section we reduce the dimensionality of the features that were retained by projecting them to a lower dimensional space. This projection can account for correlation between features, thereby reducing the number of redundant features. Numerous projection methods have been proposed, both linear and nonlinear [5]. We use a *linear* method that projects feature vectors onto directions with the largest *separability* (defined below). The projection is of the form

$$\mathbf{x} = \mathbf{P}\mathbf{y}$$

where \mathbf{y} is an n -dimensional feature vector, \mathbf{x} is the m -dimensional projected vector, and \mathbf{P} is the $m \times n$ projection matrix. The matrix \mathbf{P} is designed to maximize the separability measure defined next.

The separability measure used here is called *divergence*. The divergence, D , is defined as the difference between the expected value of the log-likelihood function given classes ω_2 and ω_1 , that is

$$D = E[l_{12}(\mathbf{x})/\omega_2] - E[l_{12}(\mathbf{x})/\omega_1] \quad (1)$$

where $E[\cdot]$ represents the expected value operator, $l_{12}(\mathbf{x})$ is the log-likelihood function, defined as

$$l_{12}(\mathbf{x}) = -\ln \left[\frac{p(\mathbf{x}/\omega_1)}{p(\mathbf{x}/\omega_2)} \right], \quad (2)$$

and $p(\mathbf{x}/\omega_i)$ is the conditional density function of \mathbf{x} (the feature vector) given the class ω_i ($i = 1, 2$). (Note that in theory $l_{12}(\mathbf{x})$ can be used as the optimal discriminator between ω_1 and ω_2 .)

The *divergence* measure of separability is similar to the Bhattacharyya distance. In fact, the two yield identical solutions in the equal covariance Gaussian problem (and nearly identical solutions when the covariances are not equal). In the *general* Gaussian problem the rows of \mathbf{P} that maximize D are determined as follows. The first row is given by

$$\mathbf{p}_1 = \frac{(\boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1})(\mu_1 - \mu_2)}{\left[(\mu_1 - \mu_2)^T (\boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1})(\mu_1 - \mu_2) \right]^{1/2}} \quad (3)$$

where μ_1 , μ_2 , Σ_1 and Σ_2 are the mean vectors and covariance matrices for classes ω_1 and ω_2 respectively. This projection capitalizes on the difference in *mean* between the two classes. Note that when $\Sigma_1 = \Sigma_2$, \mathbf{p}_1 is identical to the projection provided by Fisher's Linear Discriminant [5].

The remaining rows of \mathbf{P} capitalize on the difference in *covariance* between classes. They are chosen as the first $m - 1$ eigenvectors of the matrix $\Sigma_1^{-1}\Sigma_2$. Eigenvectors from this matrix are ordered according to the coefficients $a_i = \lambda_i + 1/\lambda_i$, where λ_i are the corresponding eigenvalues. Larger values of a_i correspond to more significant eigenvectors. The number of eigenvectors chosen for incorporation into \mathbf{P} is determined by plotting the a_i (in order, from largest to smallest) and keeping those that fall to the left of the knee of the curve. After performing this procedure on the data in this study, a total of 8 eigenvectors were kept for the Spectral features, and 12 for the LPC and Cepstral features. The overall projections, including the first component due to \mathbf{p}_1 , are summarized in the table below.

Feature Set	Input Dimension	Projected Dimension
Spectral	16	9
LPC	16	13
Cepstral	16	13

Scatter plots of the first two dimensions for each of the three projected feature sets are shown in Figures ??, ?? and ?. Note that there is significant overlap in all three cases, with the worst overlap in the Spectral features.

3.6. Bayes Error Estimates

Before proceeding with classifier design it is useful to estimate the Bayes classification error for the final feature sets. These estimates are formed in exactly the same way that they were in Section 3.4, except that in this case the estimates are based on the entire feature vector (instead of individual components). These estimates can sometimes be biased [?, ?], but they provide useful target values to strive for when designing classifiers in the next section.

Plots of the Bayes error estimates for the three feature sets are shown in Figures ??, ?? and ?. For the Spectral features the even and odd curves appear to converge near a classification error of 20%. For the LPC coefficients the curves are approaching a value between 4 and 5%. The curves for the Cepstral features are not as well behaved as in first two, suggesting that the Cepstral data may be *multi-modal*, and that the estimates formed here are probably biased high [?, ?]. At any rate, these curves suggest an error rate in the neighborhood of 15%.

As we shall see, the best classifiers for the Spectral and LPC features came close to these estimates, although both fell short by a few of percentage points. The best classifier for the Cepstral features however, performed much better than the 15% estimate obtained here (verifying that it is indeed biased high).

3.7. Classifier Design

All of the training and testing performed in this section and the next assumes that the occurrence of good and bad bearings is equally likely, i.e. the prior probability for both of these events is assumed to be 0.5. Although this is probably not true in practice, we have no knowledge of the actual priors, so we have made the worst case assumption, i.e. that the prior probabilities provide no help in discriminating between the two categories. If the true priors were known, they could easily be incorporated into the classifiers below, and would most certainly improve their performance.

The following classifiers were trained and tested on each of the three projected feature sets described

Linear: The linear classifier is the simplest and best known of all pattern classifiers. It is important to include this classifier in any type of pattern recognition study since it often provides the best solution, and even when it does not, it provides as a useful benchmark. No well-trained classifier should ever perform worse than the linear classifier on a nonlinear problem. There are an abundance of different training algorithms for the linear classifier, depending on one's objective [3, 5, 13]. Most of these attempt to optimize a specific performance criterion. Two of the most common criteria are the *least squares* (LS) and the *perceptron* (P) criterion. Minimizing the LS criterion has the advantage that it produces the optimal solution to the equal covariance Gaussian problem. On the other hand, if the data are not Gaussian, or the covariances matrices are not equal, the LS solution can be very poor. The perceptron criterion, on the other hand, works reasonably well in these situations. We trained the linear classifier using both criterion. Optimizing the LS criterion is equivalent to finding the least squares solution to a set of overdetermined linear equations, which we do using singular value decomposition. To optimize the perceptron criterion we use the *Pocket* algorithm (with Ratchet [6], which is often more efficient than the more popular *perceptron learn-*

ing algorithm found in the literature.

Quadratic: The quadratic classifier is a natural extension of the linear classifier. It is typically trained to implement the optimal solution to the Gaussian problem under the *unequal* covariance assumption. Since this is the most general version of the Gaussian problem, the quadratic classifier should always be considered when sufficient training data are available.

1-Nearest Neighbor (1-NN): The 1-NN classifier [3, 5, 13] is arguably the most popular of all traditional nonparametric classifiers. It is often referred to as a *distance classifier* [13] because its most common implementation uses *Euclidean distance* to measure “closeness” when searching for the nearest neighbor. The two major design components of a 1-NN classifier are

- determining the number of prototypes, and
- positioning the prototypes.

We use the LVQ algorithm [9] to position the prototypes. The *number* of prototypes was determined using the search procedure described below.

Radial Basis Function (RBF): The RBF network is the first of the two *neural network* classifiers considered in this study. It is often viewed as a *mixture-of-Gaussians* model. The major design components of this network include

- determining the number of Gaussians (i.e. the number of hidden layer nodes),
- determining the mean vectors and covariance matrices for each Gaussian, and
- determining the optimal weighting of the Gaussians (i.e. determining the output layer weights).

The mean vectors were determined using the K-Means algorithm [3] (which we initialized using the Maximum-Distance procedure [13]). After fixing the mean vectors, the covariance matrices were computed in the usual fashion (i.e. as the sample covariance for each cluster), except that the off-diagonal elements were set to zero. This represents a *simplification* of the full-blown mixture-of-Gaussian model, but an *extension* of the traditional RBF approach that uses covariance matrices of the form $\sigma^2\mathbf{I}$.

To train the output layer weights we used both the LS algorithm and the Pocket algorithm. The LS algorithm is the traditional choice for the RBF network, but the Pocket algorithm is likely to yield better results for classification problems.

The number of Gaussians (hidden layer nodes) was determined using the search procedure described below.

Multilayer Perceptron (MLP): The MLP is the second of the two *neural network* classifiers used in this study. We used a very conventional network with no intra-layer connections, a *tanh* activation function in the hidden layer nodes, and a *linear* activation function in the output layer node. The major design components of this network include

- determining the size of the network (i.e. the number of layers and nodes), and
- determining the weights of the network.

We used two different strategies to design the MLPs in this study. The first is a conventional approach which uses the backpropagation algorithm to learn the weights, and the search procedure described below to determine the number of nodes. The second strategy is unique to this project. It builds a 2-hidden layer network, with 10 nodes in the first hidden layer and 5 nodes in the second hidden layer, one node at a time. The 10 nodes in the first hidden layer act as linear discriminators for each of the $\binom{5}{2} = 10$ pairs of Level I groupings in Table 1 (the sixth group is ignored). That is, the first node is trained to discriminate between *New* and *Used* bearings, the second between *New* and *Ball Spall* damaged bearings, etc. Once the 10 nodes in the first hidden layer are trained, their weights are fixed and the 5 nodes in the second hidden layer are designed. Each of these nodes acts as a linear discriminator between one of the Level I groups and the other four. For example, the first node is trained to discriminate between *New* bearings and *all other* bearings, while the second is trained to discriminate between *Used* bearings and *all other* bearings, etc. Finally, with the parameters of the hidden layer nodes fixed, the output layer node is trained to discriminate between good and bad bearings. Since all nodes are trained on an individual basis, the LS or Pocket algorithm is used. The advantages of this approach are that it is much faster than

backpropagation and it eliminates the need to search for the network size.

Fuzzy ARTMAP: The ARTMAP neural network architecture is described in detail by Carpenter et al. [1]. A block diagram of this architecture is shown in Figure ??.

The network consists of two fuzzy ART (adaptive resonance theory) modules, labeled ART_a and ART_b , along with an inter-ART module. Each of the fuzzy ART modules clusters the data supplied to it according to a specific similarity metric (which makes use of the fuzzy AND operation). In a supervised learning scenario, input vectors (labeled a in Figure ??) are presented at the ART_a module, and the corresponding desired outputs (labeled O in Figure ??) are presented at the ART_b module. The purpose of the inter-ART module is to determine whether the mapping between the presented inputs and outputs is the desired one. Weights in the ART_a , ART_b , and inter-ART module are adjusted during learning to achieve the desired mapping.

A critical component in the design of the 1-NN, RBF and MLP classifiers described above is the determination of their size (i.e. the number of prototypes or nodes). The same general procedure was used to resolve this issue in all three cases. A sequential search was performed, starting with the smallest possible size and then increasing it until the classifier performance began to level off. This method works quite well in practice, and is supported by the fact that, in theory, the curve of performance verses size is unimodal.

Plots of performance verses size for the Spectral features are shown in Figures ?? – ??. Each plot shows two curves, one for *training* performance and one for *testing* performance. Note that the training and testing curves are nearly identical in all three figures. The RBF plot has two pair of curves, one for LS training and one for Pocket training. A significant improvement is obtained with Pocket training. Overall, these plots show that an increase in size yielded little or no improvement for any of the classifiers. This suggests that the *linear* classifier may be optimal for this feature set (since different types of nonlinearities don't seem to help).

A specific size (and corresponding performance) must be chosen for the 1-NN, RBF and MLP classifiers so that they can be compared with the others in the next section. Table 3.7 shows the sizes chosen

for this purpose.

Optimal Size Classifiers for Spectral Features		
Classifier	Size	% Classification Error
1-NN	6	27.77/26.09
RBF (LS)	18	33.75/34.44
RBF (Pocket)	18	27.76/27.39
MLP	10	23.79/23.73

Plots of performance verses size for the LPC features are shown in Figures ?? – ??. In this case there are significant improvements in all three classifiers as the size is increased from the minimum. Also as expected, these improvements gradually diminish as the classifier approaches its optimal size. The specific sizes chosen for each classifier are summarized in Table 3.7.

Optimal Size Classifiers for LPC Features		
Classifier	Size	% Classification Error
1-NN	38	7.00/6.91
RBF (LS)	18	27.77/28.18
RBF (Pocket)	26	19.23/20.11
MLP	30	6.90/7.54

Finally, plots of performance verses size for the Cepstral features are shown in Figures ?? – ??. Here we see improvements with the RBF and 1-NN classifiers as the size is increased from the minimum, but not with the MLP classifier. The performance of the MLP classifier starts with a much lower classification error than the other two, but shows very little improvement as its size is increased. The sizes chosen for the classifiers in this case are summarized in Table 3.7.

Optimal Size Classifiers for Cepstral Features		
Classifier	Size	% Classification Error
1-NN	26	2.43/2.46
RBF (LS)	14	16.06/16.45
RBF (Pocket)	22	2.31/2.24
MLP	10	7.09/7.56

Note that in all of the experiments with the RBF network above, the performance with the Pocket algorithm was far superior to that of the LS algorithm.

3.8. Classification Results

The classification results for the Spectral feature set are summarized in Table 2. The best classifier for this feature set is the *linear classifier trained with the Pocket algorithm*. Although one of the MLP networks achieved slightly better performance, the additional complexity of this classifier does not justify

Classifier	% Classification Error		Classifier	% Classification Error	
	Training	Test		Training	Test
Linear (LS)	24.65	24.26	Linear (LS)	8.84	9.07
Linear (Pocket)	22.07	22.63	Linear (Pocket)	6.76	6.97
Quadratic	24.09	24.56	Quadratic	5.71	6.14
1-NN (LVQ)	27.77	26.09	1-NN (LVQ)	2.43	2.46
RBF (LS)	33.75	34.44	RBF (LS)	16.06	16.45
RBF (Pocket)	27.76	27.39	RBF (Pocket)	2.31	2.24
MLP (BP)	23.79	23.73	MLP (BP)	7.09	7.56
MLP (Constructive/LS)	22.56	23.83	MLP (Constructive/LS)	3.98	3.83
MLP (Constructive/Pocket)	21.26	22.15	MLP (Constructive/Pocket)	5.06	5.33
Fuzzy ARTMAP	4.2	27.5	Fuzzy ART	4.80	9.70

Table 2: Classification Results for Spectral Features.

Table 4: Classification Results for Cepstral Features.

Classifier	% Classification Error		Run	Feature	% Good	% Bad
	Training	Test				
Linear (LS)	16.80	17.54	Run 3	Spectrum	25.2	74.8
Linear (Pocket)	15.77	16.16		LPC	85.4	14.6
Quadratic	10.18	10.78		Cepstrum	14.4	85.6
1-NN (LVQ)	7.00	6.91	Run 4	Spectrum	32.2	67.8
RBF (LS)	27.77	28.18		LPC	35.0	65.0
RBF (Pocket)	19.23	20.11		Cepstrum	60.1	39.9
MLP (BP)	6.90	7.54				
MLP (Constructive/LS)	7.40	7.45				
MLP (Constructive/Pocket)	5.65	5.86				
Fuzzy ART	2.22	31.0				

Table 3: Classification Results for LPC Features.

Table 5: Classification Results for Runs 3 and 4.

its choice given such a small increase in performance. Note that the classification performance of the linear classifier is within 3% of the Bayes estimate for this feature set.

The classification results for the LPC features are summarized in Table 3. There is wide variation in performance among the various classifiers. The best performance was achieved by the 10-5-1 MLP network which used the Pocket algorithm to train the nodes. The performance achieved by this network is within 1% of the Bayes estimate for this feature set.

The classification results for the Cepstral features are summarized in Table 4. As in the previous case, there is wide variation in performance among the various classifiers. The best overall performance was achieved by the RBF network (trained with the Pocket algorithm), with the 1-NN classifier (trained with LVQ) running a close second. The performance achieved by this network is well below the 15% Bayes error estimate for this feature set.

3.9 Tests on Unknown Bearings

In this section we describe the results of tests performed on Runs 3 and 4 whose true classifications are unknown (but believed to be “normal”). Neither of these runs were used in the training and testing above. Both runs were processed by the three different pattern recognition systems (one for each feature type). In each system the optimal classifier (determined in the previous section) was used, i.e. the linear, MLP and RBF classifiers were used for the Spectral, LPC, and Cepstral features respectively. The classification results are summarized in Table 5. Several observations can be made regarding these results.

- There is a large variation across feature sets.
- There is a large variation in the consistency of the results between Run 3 and Run 4.
- None of the results are in close agreement with the classification error rates predicted in previous sections.

The explanation for these poor results is that the data used to design these systems was not representative of all future data. This was alluded to as a

potential flaw with this approach earlier in the study. Runs 3 and 4 are significantly different (in ways that could not be predicted) from the runs used to design these systems. This is verified in the plots in Figures ?? and ?. Here we show histograms of the classifier outputs for the Spectral and Cepstral systems. A comparison is made between the output values for the training data (both Good and Bad) and the test data from Runs 3 and 4. Note that in both cases the output values for the test data have a significantly different distribution than the training data. In the Cepstral system, the outputs are essentially zero for all test inputs. This system uses an RBF classifier, which is well-known to produce zero outputs for data that is dissimilar to the training data.

These results tell us that the systems designed here are not likely to produce meaningful classifications of future data. A more promising approach, which was mentioned earlier in this report, is investigated in the next section.

4. Trending Using Fuzzy ART

We have also investigated a trending approach to the detection of faulty bearing. In order to perform a trending analysis, it is necessary to monitor the *same* bearing over an extended time period. In four of the runs supplied to us, the same bearing was used; these include Runs 13 (normal), 16 (normal), 18 (inner race), and 20 (inner race severe). Note that the last two runs correspond to damaged bearing.

Our experimental setup for the trending analysis consisting of a single fuzzy ART (clustering) module. The spectrogram data computed for Run 13 was supplied as input, and the network formed a single cluster. A histogram of the output node value for this cluster, for each of the 500 time time segments is shown in Figure ??.

Notice that the output node produced the same value for all 500 inputs.

At this point the fuzzy ART network weights were “frozen”, and the spectrogram data computed for Run 16 was supplied as input. The histogram for

Run 16 is shown in Figure ??.

This histogram is only slightly different from the previous one (i.e., it is nearly flat).

The fuzzy ART histogram corresponding to the spectrogram data computed for Run 18 is shown in Figure ??.

In this (damaged bearing) case, the histogram is significantly different from the previous two. For the final case (severe damage), Run 20, the histogram (Figure ??) is extremely “ragged”.

This experiment suggest that a trending approach, which is based on learning the features associated with a normal bearing, and monitoring these features over time, may be a feasible alternative to the other experiments performed in this study.

5. Conclusions and Future Directions

We have presented an overview of fault detection and isolation techniques with special emphasis on vibrational data and the usage of neural networks. We have presented applications of these techniques to real vibrational data. Based on this study, we have determined that the approach which presents the best probability of success is the trending approach where a particular system is monitored over its lifetime and faults are detected as deviation from normal behavior. On the other hand, an approach relying on combining data from different systems is doomed to failure as shown in section 3. As a direction of future research, we intend to study the trending approach using different feature sets and different neural networks structures.

References

- [1] G. A. Carpenter. *Neural Networks: From Foundations to Applications Short Course*, May 1991. Wang Institute of Boston University, Tyngsboro, MA.
- [2] T.P. Caudell and D.S. Newman. An adaptive resonance architecture to define normality and detect novelties in time series and databases. In *Proc. World Congress on Neural Networks*, pages IV-166-IV-176, Portland, OR, 1993.
- [3] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. Wiley, New York, NY, 1973.
- [4] P.M. Frank. Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy—a survey and some new results. *Automatica*, 26:459-474, 1990.
- [5] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, San Diego, CA, 1972.
- [6] S.I. Gallant. *Neural network learning and expert systems*. MIT Press, Cambridge, Massachusetts, 1993.
- [7] Michael J. Healy, Thomas P. Caudell, and Scott D. G. Smith. A neural architecture for pattern sequence verification through inferencing. *IEEE Transactions on Neural Networks*, 4(1):9-20, 1993.
- [8] R. Isermann. Process fault detection based on modeling and estimation methods. *Automatica*, 20:387-404, 1984.
- [9] T. Kohonen. *Self-Organization and Associative Memory*. Springer-Verlag, Berlin, Germany, 1988.
- [10] R. Kuczewski and D. Eames. Helicopter fault detection and classification with neural networks. In *Proc. Int. Joint Conf. on Neural Networks*, pages II-947-II-956, Baltimore, MD, 1992.
- [11] M. Polycarpou and A. Vemuri. “Learning methodology for failure detection and accomodation. *IEEE Contr. Sys. Mag.*, 15(3):16-24, 1995.
- [12] S.D. Stearns and D.R. Hush. *Digital Signal Analysis*. Prentice-Hall, Englewood Cliffs, NJ, 1990.
- [13] J.T. Tou and R.C. Gonzalez. *Pattern Recognition Principles*. Addison-Wesley, Reading, MA, 1974.
- [14] F.B. Weiskopf, F.G. Arcella, J.S. Lin, and R.W. Newman. A hybrid system approach to machinery monitoring and diagnosis. In *Proc. Int. Mach. Monit. & Diag. Conf.*, pages 25-30, Chicago, IL, 1993.
- [15] A.S. Willsky. A survey of design methods for failure detection in dynamic systems. *Automatica*, 12:601-611, 1976.