Genetic Algorithms for Artificial Neural Net-based Condition Monitoring System Design for Rotating Mechanical Systems

Abhinav Saxena

School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA.

Ashraf Saad, PhD

Associate Professor, School of Electrical and Computer Engineering, Georgia Institute of Technology, Savannah, GA 31407, USA.

Abstract. We present the results of our investigation into the use of Genetic Algorithms (GA) for identifying near optimal design parameters of Diagnostic Systems that are based on Artificial Neural Networks (ANNs) for condition monitoring of mechanical systems. ANNs have been widely used for health diagnosis of mechanical bearing using features extracted from vibration and acoustic emission signals. However, different sensors and the corresponding features exhibit varied response to different faults. Moreover, a number of different features can be used as inputs to a classifier ANN. Identification of the most useful features is important for an efficient classification as opposed to using all features from all channels, leading to very high computational cost and is, consequently, not desirable. Furthermore, determining the ANN structure is a fundamental design issue and can be critical for the classification performance. We show that GA can be used to select a smaller subset of features that together form a genetically fit family for successful fault identification and classification tasks. At the same time, an appropriate structure of the ANN, in terms of the number of nodes in the hidden layer, can be determined, resulting in improved performance.

Keywords: Genetic Algorithms, Artificial Neural Networks, Hybrid Techniques, Fault Diagnosis, Condition Monitoring, Rotating Mechanical Systems.

Introduction

With the increase in production capabilities of modern manufacturing systems, plants are expected to run continuously for extended hours. Consequently, unexpected downtime due to machinery failure has become more costly than ever before and therefore condition monitoring is gaining importance in industry because of the need to increase machine availability and health trending, to warn of impending failure, and/or to shut down a machine to prevent further damage. It is required to detect, identify and then classify different kinds of failure modes that can occur within a machine system. Often several different kinds of sensors are employed at different positions to sense a variety of possible failure modes. Features are then calculated to analyze the signals from all these sensors to assess the health of the system.

Significant research on fault detection in gears and bearings has been carried out so far and has resulted in the identification of a rich feature library with a variety of features from the time, frequency and wavelet domains. However, the requirements in terms of suitable features may differ depending on how these elements are employed inside a complex plant. Moreover, there are several other systems like planetary gears that are far more complex and a good set of features for them has not yet been clearly identified. Due to some structural similarities, it is natural to search for suitable features from existing feature libraries. An exhaustive set of all features that captures a variety of faults becomes very large for practical purposes. A feature selection process must be identified in order to speed up computation and to also increase the accuracy of classification. Genetic Algorithms (GA) offer suitable means to do so, given the vast search space formed by the number of possible combinations of all available features in a typical real-world application. GA-based techniques have also been shown to be extremely successful in evolutionary design of various classifiers, such as those based on Artificial Neural Networks (ANNs). Using a reduced number of features that primarily characterize the system conditions along with optimized structural parameters of ANNs have been shown to give improved classification performance (Jack and Nandi 2001, Samanta, 2004a, 2004b). The work presented herein uses GA for obtaining a reduced number of good features in addition to determining the parameters of the classifier ANN. Our results show the effectiveness of the extracted features from the acquired raw and preprocessed signals in diagnosis of machine condition.

Using Genetic Algorithms and ANNs

Feature selection is an optimization problem of choosing an optimal combination of features from a large and possibly multimodal search space. Two major classes of optimization techniques have traditionally been used, namely: calculus-based techniques, that use gradient-based search mechanisms, and enumerative techniques, such as dynamic programming (DP) (Tang *et al.* 1996). For an ill-defined or multimodal objective function where a global minimum may not be possible or very difficult to achieve, DP can be more useful but its computational complexity makes it unsuitable for effective use in most practical cases. Thus a non-conventional nonlinear search algorithm is desired to obtain fast results, and GA meets these requirements. From another standpoint the basic problem here is that of high dimensionality, with a large number of features among which it is not known which ones are good for identifying a particular fault type. There are several methods that can be used to reduce the dimensionality of the problem. Principle Component Analysis (PCA) is the most widely used technique among similar techniques such as Multi Dimensional Scaling (MDS) and Singular Value Decomposition (SVD) (Vlachos *et al.* 2002). Other methods are low dimensional projection of the data (projection pursuit, generalized additive models) (Miguel 1997), regression (principle curves), and self-organization (Kohonen maps) just to name a few. PCA can be much less computationally expensive than a GA-based approach. However, all features need to be computed for PCA before a rotated feature space can be created and therefore it still requires computation of all features demanding a large amount of data processing. GA facilitates a better scenario, in which although the computational cost will be very high during the offline training and feature selection phase, much less computing is required for online classification. Other methods for feature selection include Forward Selection that assesses almost all combinations of different features that are available to determine a set of best features (Dallal 2004). The main difficulty in applying Forward Selection occurs when two features individually perform poorly, but lead to better results when used together. The pair of such good features found by Forward Selection may not necessarily be the best combination, as it chooses the candidates based on their respective individual performances. GA-based methods can be used to add more functionality in parameter selection. For instance, it can be used to simultaneously find the optimal structure of an ANN, in terms of concurrently determining the number of nodes in the hidden layers and the connection matrices for evolving the ANNs

(Balakrishnan and Honavar 1995, Edwards *et al.* 2002, Filho *et al.* 1997, Sunghwan and Dagli 2003). Potential applications of ANNs in automated detection and diagnosis have been shown in (Jack and Nandi 2001, Samanta 2004a, Shiroishi e*t al.* 1996). Similar optimization can be considered irrespective of what classification technique is used, an evolving search is expected to provide a better combination, especially in the cases where the dimensionality increases the possible number of combinations exponentially, and hence the computational power needed. Approaches using Support Vector Machines and Neuro-Fuzzy Networks have also been put forward for solving the feature selection problem (Samanta 2004a), but the use of GA still remains warranted.

Genetic Algorithms

In 1975, Holland introduced an optimization procedure that mimics the process observed in natural evolution called Genetic Algorithms (Holland 1975). GA is a search process that is based on the laws of natural selection and genetics. As originally proposed, a simple GA usually consists of three processes *Selection, Genetic Operation* and *Replacement.* Figure 1 shows a typical GA cycle.

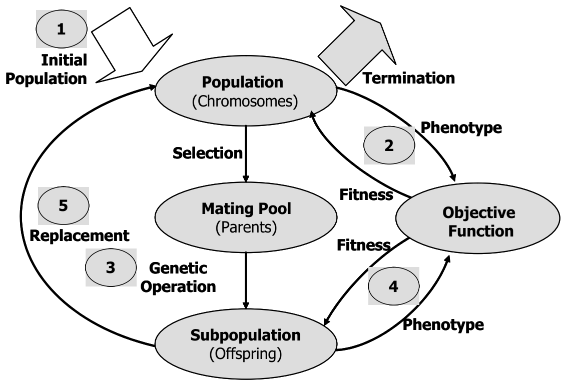


Fig. 1. Genetic algorithm life-cycle

The population comprises a group of chromosomes that are the candidates for the solution. The fitness values of all chromosomes are evaluated using an objective function (performance criteria or a system's behavior) in a decoded form (phenotype). A particular group of *parents* is selected from the population to generate *offspring* by the defined genetic operations of *crossover* and *mutation*.

The fitness of all offspring is then evaluated using the same criterion and the chromosomes in the current population are then replaced by their offspring, based on a certain replacement strategy. Such a GA cycle is repeated until a desired termination criterion is reached. If all goes well throughout this process of simulated evolution, the best chromosome in the final population can become a highly evolved solution to the problem.

Artificial Neural Networks

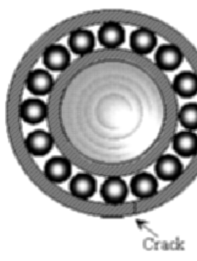
An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. A great deal of literature is available explaining the basic construction and similarities to the biological neurons. The discussion here is limited to a basic introduction of several components involved in the ANN implementation. The network architecture or topology (including: number of nodes in hidden layers, network connections, initial weight assignments, activation functions) plays a very important role in the performance of the ANN, and usually depends on the problem at hand. In most cases, setting the correct topology is a heuristic model selection. Whereas the number of input and output layer nodes is generally suggested by the dimensions of the input and the output spaces. Too many parameters lead to poor generalization (over fitting), and too few parameters result in inadequate learning (under fitting) (Duda *et al.* 2001). Some aspects of ANNs are described next.

Every ANN consists of at least one *hidden layer* in addition to the input and the output layers. The number of hidden units governs the expressive power of the net and thus the complexity of the decision boundary. For well-separated classes fewer units are required and for highly interspersed data more units are required. The number of synaptic weights is based on the number of hidden units representing the degrees of freedom of the network. Hence, we should have fewer weights than the number of training points. As a rule of thumb, the number of hidden units is chosen as *n/10*, where *n* is the number of training points (Duda *et al.* 2001, Lawrence *et al.* 1997). But this may not always hold true and a better tuning might be required depending on the problem.

Network Learning: Input patterns are exposed to the network and the network output is compared to the target values to calculate the error, which is corrected in the next pass by adjusting the synaptic weights. Several training algorithms have been designed; the most commonly used being the Levenberg-Marquardt (LM) back-propagation algorithm which is an extension of LMS algorithms for linear systems. However, resilient back propagation has been shown to work faster on larger networks (Riedmiller 1993), and has thus been used throughout this study. Training can be done in different ways. In *Stochastic Training* inputs are selected stochastically (randomly) from the training set. *Batch Training* involves presenting the complete training data before weights are adjusted, and thus several epochs are required for effective training. For *Incremental Training* weights are adjusted after each pattern is presented. ***Stopping Criterion*** indicates when to stop the training process. It can be a predetermined limit of absolute error, Minimum Square Error (MSE) or just the maximum number of training epochs.

Problem Description and Methods

A simpler problem of a roller bearing health monitoring has been used to illustrate the effectiveness of GA in feature selection for fault classification using ANNs. Several bearings with different outer race defects (Figure 2) were used in the test setup.



Experimental Setup

Fault Conditions: 9 bearings (8 with varying groove sizes to simulate growing crack + 1 without crack). 8 cracks categorized into three risk categories low(1), medium(2), high(3) and the healthy bearing formed category 4.

Data Channels: 4 sensors (3 accelerometers + 1 Acoustic Emission (AE) sensor); i.e. data is 4-D

Experiments: 9 experiments (for 8 faulty bearings + 1 healthy bearing)

Data Acquisition: *(Using LabView)* Done at 50KHz for ~5.24 sec to yield 2^{18} sample points per experiment.

Data Processing: *(Using Matlab)* Data per experiment was divided in 128 segments of 2048 samples each.

Feature Extraction: Up to 242 features were calculated for each segment from all experiments, yielding 128 x 9 feature vectors.

Data Partition: 128 segments divided into 40:60 ratios for training and testing.

Fig. 2. (a) Outer race roller bearing defect. (b) Experimental Setup

Defects were constructed using a die sinking Electrical Discharge Machine (EDM) to simulate 8 different crack sizes resulting in 9 health conditions; 8 faulty and one healthy. The groove geometries used in these tests have been described in (Billington 1997). In all, four sensors (three accelerometers and one acoustic emission sensor) were employed to get the signal from all eight faulty bearings plus one bearing without any defect. These sensors were attached to signal conditioners and a programmable low pass filter such that each had ground-isolated outputs. The defect sizes were further categorized in four groups including a no-defect category. All tests were run at a radial load of 14730 N and a rotational speed of 1400 RPM. The data was acquired in the time domain as several snapshots of the time series for each case.

Feature Extraction

As mentioned earlier, published research has been conducted on diagnostic feature extraction for gears and bearing defects. Statistical features provide a compact representation of long time series data. A great deal of information regarding the distribution of data points can be obtained using first, second and higher order transformations of raw sensory data, such as various moments and cumulants of the data. Features such as mean, variance, standard deviation, and kurtosis (normalized fourth central moment) are the most common features employed for rotating mechanical components. It is also worthwhile sometimes to explore other generalized moments and cumulants for which such common names do not exist. An approach suggested in (Jack and Nandi 2001) has been adapted to create a variety of such features in a systematic manner. Four sets of features were obtained based on the data preprocessing and feature type. These features were calculated from the data obtained from all sensors for all fault levels.

1) *Plain statistical features:* A number of features were calculated based on moments ($m_x^{(i)}$) and cumulants ($C_x^{(i)}$) of the vibration data obtained from the sensors. Where the i^{th} moment $m_X^{(i)}$ of a random variable X is given by the expectation $E(X^i)$ and the cumulants $C_x^{(i)}$ are defined as described below in equations 1-4:

$$C_x^{(1)} = m_x^{(1)} \tag{1}$$

$$C_x^{(2)} = m_x^{(2)} - (m_x^{(1)})^2 \tag{2}$$

$$C_x^{(3)} = m_x^{(3)} - 3\, m_x^{(2)} m_x^{(1)} + 2(m_x^{(1)})^3 \tag{3}$$

$$C_x^{(4)} = m_x^{(4)} - 3(m_x^{(2)})^2 - 4\, m_x^{(3)} m_x^{(1)} + 12 m_x^{(2)} (m_x^{(1)})^2 - 6(m_x^{(1)})^4 \tag{4}$$

$$w = \bullet(x^2 + y^2 + z^2) \tag{5}$$

$$\begin{aligned}
\underline{v} = [\quad & m_x^{(1)}\ m_y^{(1)}\ m_z^{(1)} \\
& C_x^{(2)}\ C_y^{(2)}\ C_z^{(2)}\ C_x^{(1)}{*}C_y^{(1)}\ C_y^{(1)}{*}C_z^{(1)}\ C_z^{(1)}{*}C_x^{(1)}\ C_x^{(3)}\ C_y^{(3)}\ C_z^{(3)}\ C_x^{(1)}{*}C_y^{(2)}\ C_x^{(2)}{*}C_y^{(1)} \ldots \\
& C_y^{(1)}{*}C_z^{(2)}\ C_y^{(2)}{*}C_z^{(1)}\ C_z^{(1)}{*}C_x^{(2)}\ C_z^{(2)}{*}C_x^{(1)} \\
& C_x^{(4)}\ C_y^{(4)}\ C_z^{(4)}\ C_x^{(1)}{*}C_y^{(3)}\ C_x^{(2)}{*}C_y^{(2)}\ C_x^{(3)}{*}C_y^{(1)} \ldots \\
& C_y^{(1)}{*}C_z^{(3)}\ C_y^{(2)}{*}C_z^{(2)}\ C_y^{(3)}{*}C_z^{(1)} \ldots
\end{aligned}$$

$$C_z^{(1)}{*}C_x^{(3)}\ C_z^{(2)}{*}C_x^{(2)}\ C_z^{(3)}{*}C_x^{(1)}$$
$$m_w^{(1)}\ C_w^{(2)}\ C_w^{(3)}\ C_w^{(4)}$$
$$m_a^{(1)}\ C_a^{(2)}\ C_a^{(3)}\ C_a^{(4)} \qquad]^T \tag{6}$$

Three accelerometers were used to measure the individual vibration components in three directions x, y and z. In order to compute the combined effect another signal *w* was generated (Equation 5). Cumulants, as described in Equations 1-4 were computed for all *x, y, z, w,* and *a* (acoustic emission) signals, and a 38 element feature vector was formed as described by Equation 6. In total, 128 snapshot segments from each of the nine experiments were used to form a sequence of 1152 (128x9) sample points. Each sample point consists of four rows of time series data from the four sensors, and the subsequent feature calculation yields a 38-element feature vector as determined by Equation 6. Thus a 1152x38 feature matrix was obtained for conducting the experiments with statistical features.

2) *Signal Differences and Sums:* As a further preprocessing on raw data, sum and difference signals were calculated for all 4 sensor channels in order to highlight high and low frequency content of the raw signals, as shown in Equations 7 and 8. Difference signals should increase whenever the high frequency changes take place, and the sum signals would show similar effects for the low frequency content.

$$d(n) = x(n) - x(n\text{-}1) \tag{7}$$

$$i(n) = \{\, x(n) - m_x^{(1)} \,\} + i(n\text{-}1) \tag{8}$$

where $m_x^{(1)}$ is the mean of the sequence *x.*

Equations 1-6 were applied to both *d(n)* and *i(n)* to create two more 38x1152 feature matrices. Since these sum and difference signals consist of only first order derivatives, computing them is fairly fast for the online processing requirements, and hence are suitable candidates as signals for feature calculation. Moreover several sensors are capable of computing these measurements internally (using hardware filtering) thereby eliminating the requirement of further data processing.

3) *Spectral Features:* Frequency domain features are very informative for rotating components, since well- defined frequency components are associated with them. Any defect associated with the balls or the inner race of bearings expresses itself as a high frequency component. For each of the four channels, a 64-point Fast Fourier Transform (FFT) was carried out. Based on visual inspection, or simple threshold test, it was observed that the frequency components were relatively small in magnitudes beyond the first 32 values. Consequently, only the first 32 values from each channel

(i.e., a total of 128 values from all four channels) were retained. Although this could be done automatically using some preset threshold, a fixed value was used for this study. This results in another 128x1152 feature matrix for spectral features. Although other features can be computed based on FFT of the raw signal, these were considered sufficient in order to show the effectiveness of GA in selecting the best features among the ones available.

Five feature sets were defined: 1) statistical features on raw signal (38 values); 2) statistical features on sum signal (38 values); 3) statistical features on difference signals (38 values); 4) spectral features (128 values), and 5) all the features considered together (242 values). The raw data was normalized using Equation 9 prior to the feature calculation. It has been shown that the normalized data performs much better in terms of training time and training success (Jack and Nandi 2001**.** This helps in fast and uniform learning of all categories and results in small training errors (Duda *et al.* 2001).

$$x_i = (x_i - \mu_x)/\ _x \qquad (9)$$

Where μ_x is the mean and $\ _x$ is the variance of the sequence x

As can be realized, a large feature set was obtained with only limited feature calculation techniques and a sensor suit mounted only at one location. In practice several such sensor suites are mounted at different locations to monitor for the occurrence of various faults, and this further increases the dimensionality of the problem. Almost all ANN-based classification techniques would take a long time to train such large networks and may not still achieve a good performance. Forming all combinations of a reduced feature set and testing them exhaustively is practically impossible. Further, searching for an optimal structure of the ANN increases the complexity of the search space considerably. In such a vast search space GA is the most appropriate non-linear optimization technique. The details of implementation for this research are described below.

Implementation

The experiments were conducted using an actual test bed and the computer implementation was done using MATLAB ver.6.5 on a PC with Pentium Celeron 2.2 GHz processor and 512 MB RAM.

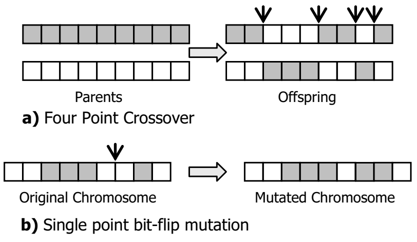
Chromosome encoding: A binary chromosomal representation was adopted. The length of the chromosome depends directly on the number of features required in the solution set for inputs to the ANN. For this study, this number was fixed at 10. This number is usually defined by the amount of time and computational power available. However, a GA-based search augmented with additional constraints and cost functions for fitness evaluation can be used to dynamically find an optimal value for the number of features to use, rather than pre-specifying it explicitly. Since the selection should be made from 38 (76, 128 or 242 depending on the feature set used) values, each gene consists of 6 (7, 7 or 8) bits (respectively). As a result, the total length of the chromosome becomes 60 (6x10) and likewise (70, 70 or 80, respectively) in other cases. In order to take care of genotypes without meaningful phenotypes, the numbers generated were wrapped around the maximum number of features in the corresponding feature set. Several experiments were carried out in order to find an appropriate population size using trial and error. It was found that a population size of 20 was appropriate for convergence in reasonable time and was therefore kept fixed at 20 for all of the experiments. Later it was realized that an integer coding of size ten could be a faster and more efficient method and also take care of genotypes without meaningful phenotypes.

Crossover: We implemented a multi-point crossover. The location of crossover points is determined randomly. All members of the population were made to crossover after they were paired based on their fitness. The chromosomes are then sorted in the order of decreasing fitness, and pairs were formed with immediate neighbors; i.e., the best chromosome was paired with the next best and the third best chromosome with the next one on the list, and so on. Figure 3a illustrates a 4-point crossover.

Mutation: A multipoint bit-flip mutation based on a pre-specified probability of mutation (p_m) was implemented. Again the location of mutation is randomly determined every time the mutation operator is applied to a chromosome. Figure 3b illustrates a single point mutation.

Fitness Function: Two fitness functions were used to capture the system performance. The number of correct classifications by the ANN was used as the first fitness function. The output of the ANN was expected to be 1, 2, 3 or 4 in value corresponding to one of four different fault levels, based on which fault level is recognized. The ANN output was rounded to the nearest integer to make a decision as shown in Figure 3c. This represents the classification success of the ANN and rejects the magnitude of training error. However, in order to evaluate the performance in terms of training efficiency, the second criterion used is the total absolute error. The total absolute error represents the accuracy of the ANN training. No direct penalty was made for incorrect classifications. However, other fitness functions can also be implemented that might involve mean squared error or some other penalty scheme over incorrect classifications.

After a new generation of offspring is obtained, the fitness of all chromosomes (both parents and offspring) is evaluated, and the ones with the highest fitness are carried to the next generation for the next genetic cycle. Two experiments were conducted in order to compare the performance of standalone ANNs and GA-supported ANNs.



Total Absolute Error
Summation of the difference between the ANN output and Actual target value for each data point
$\Sigma \mid z(i) - t(i) \mid$; $z(i)$ = ANN output, $t(i)$ = target value

Number of Correct Classifications
If $round\{z(i)\} - t(i)\} = 0$; consider it a hit.
Count all the hits and return as fitness value

c) Fitness evaluation criteria

Fig. 3. Basic building blocks and design parameters in GA implementation

First, the comparison was made between the performance of a fixed size ANN with and without GA optimization. Thus an ANN with all features used as input is compared to an ANN where the 10 best features selected by GA are used as input. The number of hidden nodes was kept fixed at 12 nodes in both cases. It was found that in most cases the best ANN performance without GA optimization was achieved with 12 hidden nodes. The size of standalone ANNs is significantly larger with several input nodes than the GA supported ANNs having only ten inputs. In the next comparison, the number of hidden nodes is also evolved with GA, and thus the size of ANN further changed every time a new chromosome is obtained. Each ANN is trained for 50-100 epochs before using the test data to measure its performance. To limit the computational time to reasonable extents, the stopping criterion was preset to the number of epochs, since each generation required training 200 ANNs, and in each case the GA is allowed to evolve over 100 generations. A resilient back-propagation training algorithm has been shown to work faster on larger networks by Riedmiller and Braun (1993), and has been used throughout this study. A batch training strategy is employed with *tan-sigmoid* activation functions for the input and hidden layers and a pure-linear activation function for the output layer.

Results and Discussion

Experiments were carried out with constant crossover and mutation parameters for result comparisons between evolved ANN and fixed sized ANN. All results have been summarized in Tables 1 and 2. The numbers in the first column represent the data set. The next columns describe the training conditions, represented by alphabets explained below. In table 2 an extra column shows the number of hidden nodes (n), GA converged to. Figure 4 shows the resulting convergence curves for both fitness functions.

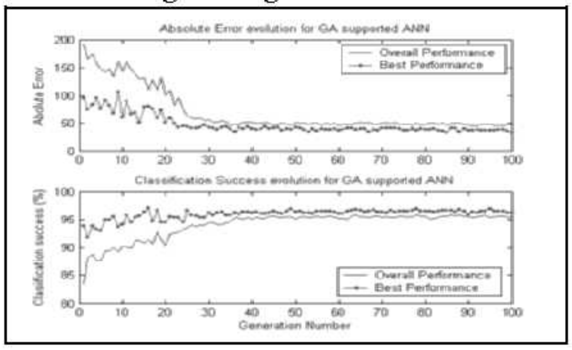


Fig. 4. GA convergence curves for the two fitness functions.

It can be seen that in all cases GA supported ANNs clearly outperform the standalone ANNs. In some cases; for instance when using features on Sum signal (dataset 2), the ANN completely failed because the feature values were zero in many cases due to very small low-frequency content change in the signal. That in turn diminishes the statistical features computed based on it. GA, however uses only the non-zero features to train the ANN and thereby showing a significantly better performance. In case 5 (including case 2 data) ANNs do not train very well, and once again GA-ANNs result in superior performance. Table 2 compares the performance of standalone ANNs with GA evolved ANNs (with optimized number of hidden nodes). The number of hidden nodes converged to 7 in most cases. For the dataset used in this study, it can be seen that statistical features give the best performance, with minimum absolute errors and maximum classification success. However, when all features are used together (dataset 5) to search for the best solution, the performance further improves, and this could be a direct result of the fact that good features from other datasets give the best performance in conjunction with good statistical features. Since the training times for all these different cases were too high, only 50 epochs were used in most cases, sufficient to show the resulting improvement upon using the GA. Overall, it can be concluded that GA-evolved ANNs perform much better, and can lead to considerable savings of computational expenses for effective bearing health monitoring. Therefore, once these optimal parameters have been found, such a system can be used for online classification of the faults. In that case we expect to use a select feature set and a fixed ANN structure to identify and classify the fault. Such optimal parameters can be scheduled for re-computation to take into account the non-stationary nature of the process. These kinds of calibrations can be done concurrently offline and employed online once the new parameters are available.

Table 1. Comparing performance of standalone ANNs with GA supported ANN

Training Conditions					**Min Absolute Error**		**Mean Absolute Error**		**Best Classification Success (%)**		**Mean Classification Success (%)**	
D†	E/G/P/C/M *				ANN-12	GANN-12	ANN-12	GANN-12	ANN-12	GANN-12	ANN-12	GANN-12
1	100	100	20	5/3	84.90	31.9	131	47.9	97	100	92	98.2
2	75	100	20	5/1	815	32.6	816	62.2	22.2	88	22	82
3	50	100	20	5/3	97.85	64.99	115.3	94.79	98.5	99.67	97.2	97.6
4	50	100	20	5/3	82.11	57.27	601.1	114.45	95.1	98.1	65.22	95.5
5	50	100	20	5/3	2048	18.42	2048	39.63	22	100	22	99.65

Table 2. Comparing the performance of fixed size ANNs with GA evolved ANNs.

Training Conditions			**Min Absolute Error**		**Mean Absolute Error**		**Best Classification Success (%)**		**Mean Classification Success (%)**	
D†	E/G/P/C/M*	n	ANN-12	GANN-n	ANN-12	GANN-n	ANN-12	GANN-n	ANN-12	GANN-n
1	50/50/20/5/5	7	84.90	12.12	131	22.53	97	100	92	99.46
2	50/50/20/5/5	7	815.00	111.8	816	155.8	23	99.22	22	96.58
3	50/50/20/5/5	8	97.85	45.76	115.3	87.12	98.5	100	97.2	99.3
4	50/50/20/5/5	8	82.11	91.35	601.1	115.16	95.1	95.49	65.22	93.5
5	50/50/20/5/5	7	2048	11.9	2048	23	22	100	22	99.94

† **1**: Statistical Features (38), **2**: Sum features (38), **3**: Difference features (38), **4**: Spectral Features (128), **5**: All the features (242) * D: Data Set, E: # epochs for ANN training, G: Number of generations, P: Population size, C: crossover points, M: mutation points, *n*: Number of hidden node

Conclusion and Future Work

It has been shown that using GAs for selecting an optimal feature set for a classification application of ANNs is a very powerful technique. Irrespective of the vastness of the search space, GA can successfully identify a required number of good features. The definition of "good" is usually specific to each classification technique, and GA optimizes the best combination based on the performance obtained directly from the success of the classifier. The success of the ANNs in this study was measured in both terms; i.e., the training accuracy and the classification success. Although the end result depends on the classification success, the training accuracy is the key element to ensure a good classification success. Having achieved that objective, further scope of this research is in two directions. First, to make the ANN structure more adaptive so that not only the number of hidden nodes but also the number of input nodes and the connection matrix of the ANNs can be evolved using GA. Second, to use this technique to find out a desired set of good features for more complex systems such as planetary gear systems, where some faults cannot be easily distinguished.

Acknowledgments

The authors would like to thank the Woodroof School of Mechanical Engineering, and the Intelligent Control Systems Laboratory at the School of Electrical and Computer Engineering, the Georgia Institute of Technology, for the experimental data sets obtained for the work.

References

Billington S A (1997), 'Sensor and Machine Condition Effects in Roller Bearing Diagnostics", *Master's Thesis, Department of Mechanical Engineering*, Georgia Institute of Technology, Atlanta.

Balakrishnan K, Honavar V (1995), 'Evolutionary Design of Neural Architecture – A Preliminary Taxonomy and guide to Literature", *Artificial Intelligence Research group, Iowa State University, CS Technical Report #95-01.*

Drakos N, 'Genetic Algorithms as a Computational Tool for Design" http://www.cs.unr.edu/~sushil/papers/thesis/thesishtml/thesishtml.html

Dallal G E (2004), 'The Little Handbook of Statistical Practice", http://www.tufts.edu/~gdallal/LHSP.HTM.

Duda R O, Hart P E, Stork D G (2001), *Pattern Classification* , Second Edition, Wiley-Interscience Publications.

Edwards D, Brown K, Taylor N(2002), "An Evolutionary Method for the Design of Generic Neural Networks", CEC '02. *Proceedings of the 2002 Congress on Evolutionary Computation*, vol. 2 , pp. 1769 –1774.

Filho E F M, de Carvalho A (1997), 'Evolutionary Design of MLP Neural Network Architectures", *Proceedings of IVth Brazilian Symposium on Neural Networks*, pp. 58 - 65

Goldberg D E (1989), *Genetic Algorithm in Search, Optimization and Machine Learning*, Addison Wesley Publishing Company

Holland J (1975), *Adaptation In Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor

Jack L B, Nandi A K (2000), 'Genetic Algorithms for Feature Selection in Machine Condition Monitoring With vibration Signals", *IEE Proceedings, Image Signal Processing*, Vol. 147(3), June, pp 205-212.

Lawrence S, Giles C L, Tsoi A C (1997), 'Lessons in Neural Network Training: Overfitting May be Harder than Expected", *Proceedings of the Fourth National Conference on Artificial Intelligence*, AAAI-97, pp 540-545.

Miguel ÁCarreira-Perpiñán (1996), *A Review of Dimension Reduction Techniques* (1997), Dept. of Computer Science, University of Sheffield, technical report CS-96-09.

Riedmiller M, Braun H (1993), "A direct adaptive method for faster backpropagation learning: the RPROP algorithm", *IEEE International Conference on Neural Networks*, vol. 1, pp. 586 –591.

Samanta B (2004a), 'Gear Fault Detection Using Artificial Neural Networks and Support Vector Machines with Genetic Algorithms", *Mechanical Systems and Signal Processing*, Vol. 18, pp. 625-644.

Samanta B (2004b), "Artificial Neural Networks and Genetic Algorithms for Gear Fault Detection", *Mechanical Systems and Signal Processing* (Article in press).

Shiroishi J, LiY, Liang S, Kurfess T, Danyluk S (1997), 'Bearing Condition Diagnostics via Vibration & Acoustic Emission Measurements", *Mechanical Systems and Signal Processing*, 11(5

Sunghwan S, Dagli C H (2003), Proceedings of the International Joint Conference on Neural Networks, vol. 4, pp. 3218 –3222.

Tang K S, Man K F, Kwong S, He Q (1996), 'Genetic Algorithms and Their Applications", *IEEE Signal Processing Magazine* (11), pp 21-37

Vlachos M, Domeniconi C, Gunopulos G, Kollios G (2002), 'Non-Linear Dimensionality Reduction Techniques for Classification and Visualization", *Proceedings of 8th SIGKDD* Edmonton, Canada.