Available online at www.sciencedirect.com

ScienceDirect

Journal of Sound and Vibration 309 (2008) 529–544

JOURNAL OF SOUND AND VIBRATION

www.elsevier.com/locate/jsvi

Genetic optimisation of a neural damage locator

K. Worden*, G. Manson, G. Hilson[1], S.G. Pierce[2]

Department of Mechanical Engineering, University of Sheffield, Mappin Street, Sheffield S1 3JD, UK

Received 8 June 2006; received in revised form 15 July 2007; accepted 19 July 2007
Available online 18 September 2007

Abstract

A critical problem in structural health monitoring (SHM) based on pattern recognition methods is the correct selection of features, i.e. measured and processed data for the diagnosis. Various selection strategies have been applied in the past and one approach that has proved effective is the use of combinatorial optimisation methods. This paper presents a case study based on a scheme for damage location in an aircraft wing. The feature selection algorithm is a Genetic Algorithm and the locator (classifier) is an artificial neural network. A comparison is made with the results obtained when the features are selected on the basis of engineering judgement. The study is seen to raise some issues relating to model complexity and generalisation and these matters are discussed in some detail.

1. Introduction

This paper is the latest in a series on the experimental validation of vibration-based structural health monitoring. It follows Rytter's idea [1] that SHM is a hierarchical process starting with damage detection and moving through location and quantification to residual life prediction. The first two papers in the series were concerned with detection, and made use of outlier analysis in order to flag faults in a laboratory structure [2], and an aircraft wing [3]. The third paper in the series is concerned with the problem of damage location in an aircraft wing [4], and the most recent paper considered the problem of severity assessment in the same structure [5].

The current paper returns to the problem of damage location. In Ref. [4], this was accomplished by training a neural network classifier on vibration-based features. Although the feature-selection process was informed by some statistical analysis, it was not fully objective and could not be considered optimal. The objective of this study is to investigate an optimisation-based approach to the feature selection in order to see if the classification (location) results can be improved.

*Corresponding author. Tel.: +44 114 222 7758; fax: +44 114 222 7890.
E-mail address: k.worden@sheffield.ac.uk (K. Worden).
[1]Current address: Department of Physics, University of Bristol, UK.
[2]Current address: Department of Electronic and Electrical Engineering, Centre for Ultrasonic Engineering (CUE), University of Strathclyde, Royal College Building, 204 George Street, Glasgow G1 1XW, UK.

0022-460X/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jsv.2007.07.035

Because the problem is one of combinatorial optimisation, the Genetic Algorithm (GA) is ideally suited. It has already proved its worth in the SHM context in the optimisation of sensor networks [6,7], and in the context of condition monitoring [8].

The layout of the paper is as follows: the second section is concerned with describing the experimental structure — a Gnat trainer aircraft — and the acquisition of the data. The third section outlines how features were selected in the previous work and how they were used to train a neural network damage locator. The next section describes the GA approach and presents the results. Section Five discusses issues relating to generalisation of the neural network models and the paper is rounded off with brief conclusions.

2. Test set-up and data capture

As discussed above, the structure under investigation was a Gnat trainer aircraft with the object of the exercise being to locate damage in the starboard wing (Fig. 1). As it was undesirable to permanently damage the wing structure, damage was simulated by sequentially removing inspection panels; this had the advantage that each damage scenario was reversible and it was possible therefore to monitor the repeatability of the measurements.

Of the various panels available, nine were chosen, mainly for their ease of removal and also to cover a range of sizes. These were distributed as shown in Fig. 1. (Fig. 1 is not drawn to scale, it is a schematic showing a rough estimate of the relative sizes of the panels, the actual panel areas are given in Table 1). Removal of any of the panels actually constitutes a rather large damage. Panels P3 and P6 were deemed likely to give the most difficulty as they are by far the smallest. In fact the areas of the smaller panels are comparable with the sizes of battle damages inflicted on military aircraft in combat.

Each panel was fixed to the wing by a number of screws, the numbers varying between 8 and 26. The screws were secured and removed with an electric screwdriver with a controllable torque. The same torque setting was used throughout in an attempt to control variability in the fixing conditions.

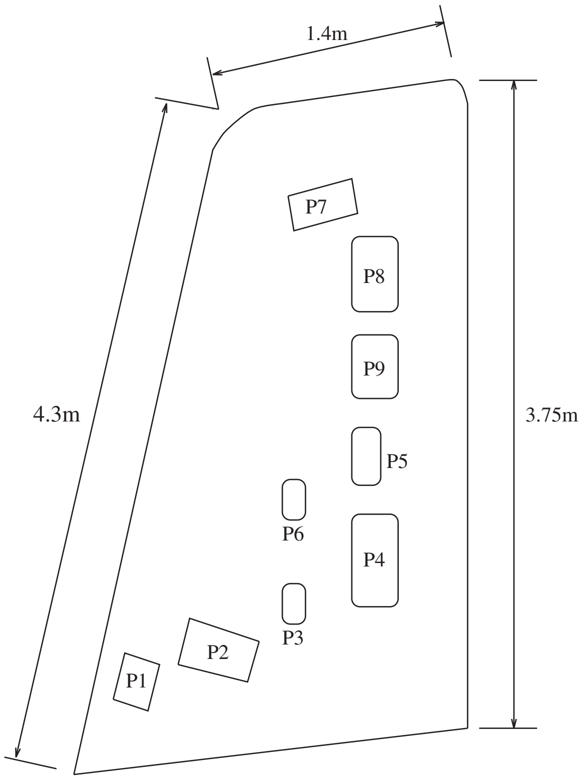

Fig. 1. Schematic of wing showing panel locations.

Table 1
Area of inspection panels

Panel	Area (m^2)
1	0.0221
2	0.0496
3	0.00825
4	0.08
5	0.0176
6	0.00825
7	0.0392
8	0.0468
9	0.0234

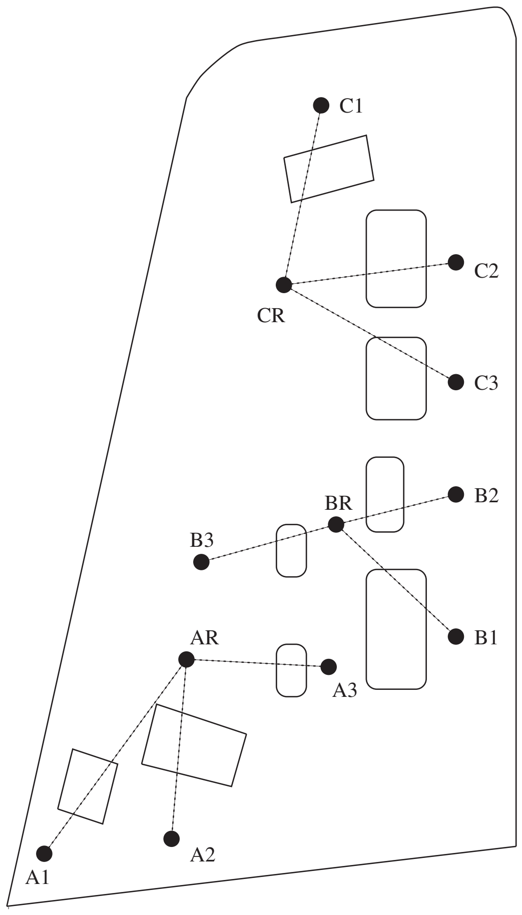

Fig. 2. Schematic of wing showing transducer groups.

Transmissibilities were used for the base measurements, the reasons for this are discussed in more detail in Refs. [2–4]. Transmissibilities across each panel were measured. The panels were split into three groups A, B and C. Each group was allocated a centrally placed reference transducer, together with three other transducers, each associated with a specific panel. The transducer layout is shown in Fig. 2. In this figure, for example, the symbol AR refers to the reference transducer for group A and A1, A2 and A3 are the response transducers associated with each of the three panels in the group. A transmissibility was associated with each response transducer by taking the ratio of the acceleration response spectrum and the reference acceleration spectrum. In all cases, the spectra were estimated using standard Fast Fourier Transforms with appropriate windowing and averaging. There is no suggestion that the placing of the transducers here is optimal. The locations were simply chosen in order that the locale of *each* panel was covered by at least one sensor pair. In principle, the locations of the sensor pairs could be included in the overall optimisation considered here; however, it was decided to concentrate on feature selection alone with a fixed sensor distribution.

Although this network of sensors offered the possibility of forming many transmissibilities, only those measured directly across the panels were used in the study. The transmissibility across panel P*n* was denoted T*n*. (Later when feature selection is discussed, the reciprocals of these transmissibilities will be denoted T*n**.)

The wing was excited using a Ling electrodynamic shaker attached directly below panel P4 on the bottom surface of the wing. A white Gaussian excitation was generated within the acquisition system and amplified using a Gearing and Watson power amplifier.

The transmissibilities were measured using a DIFA Scadas 24-channel acquisition system controlled by LMS software running on a HP workstation. In all cases 1024 spectral lines were measured between the frequencies 1024 and 2048 Hz; both real and imaginary parts of the functions were obtained; however, these were converted to magnitudes for feature selection — the phases were discarded. With hindsight, the phases should have been saved as a possible source of useful features, a new experimental programme on a different wing is currently in progress to consider the use of phase measurements.

In total, 25 sets of measurements were made, two sets for each damage state (specific panel removal) and seven for the normal condition (no panels removed). The detailed test schedule is given in Ref. [4]. In all cases, each transmissibility was first obtained using 16 averages (to provide a clean reference to help with feature selection). Next, 100 measurements were taken sequentially using only 1 average. (In practice there will be a trade-off between signal-to-noise reduction due to averaging and speed of acquisition and computing. In this case, single averages were taken as a 'worst-case' in order to investigate the worst-case sensitivity of the algorithms.) Over the full sequence of 25 configurations, this gave 700 one-shot measurements for the normal condition and 200 for each damage condition.

3. Feature selection, novelty detection and damage location

The first stage in the original analysis [4], was to establish which features could be used to individually detect damage in the panels. The detection algorithm used was the outlier analysis procedure described and validated in Refs. [2,3]. The idea of the method is to construct a statistical model of the normal data and test subsequent data to see if they are statistically consistent with the normal model. Inconsistent data vectors are taken to infer damage. There is an implicit assumption here that inconsistency with normal condition can *only* arise from damage. In fact, novelty in the data could, in principle, arise from changes in the operational or environmental conditions since the normal condition data were measured. This is a difficult problem, one of the most pressing in the field of SHM, a good survey of current research is given in Ref. [9].

For the purposes of the study, a feature was taken as a region of a given transmissibility which separates unambiguously the normal condition data from the damage data. Fig. 3 shows a feature which was selected for its ability to signal the removal of P7.

In order to explain the selection procedure, a little terminology is necessary. As this phase of the work entailed the visual examination of many transmissibilities and the number of candidate features was very large, in order to separate out the best in as objective a manner as possible, features were classified as *weak*, *fair* or *strong* according to the following criteria:

- A **strong** feature is a region of the frequency range on which the normal data and damage data appear to be structurally different. Also, the damage data should be strongly separated compared to the spread of the normal data.
- A **fair** feature is a region of the frequency range on which the normal data and damage data appear to be structurally different *or* the damage data are strongly separated compared to the spread of the normal data.
- A **weak** feature is a region of the frequency range on which the normal data and damage data are separated.

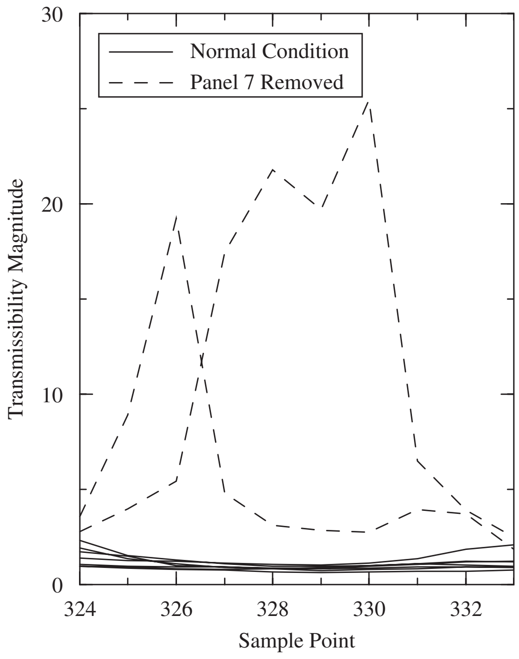

Fig. 3. Typical transmissibility based feature (strong). The solid line shows samples from the transmissibility over a given frequency range for data acquired from the normal condition, while the dashed line shows the same frequency range of the transmissibility for data acquired when Panel 7 was removed.

There is, of course, still an element of subjectivity associated with this process — the 'engineering judgement' referred to in the abstract comes into play here, e.g. the authors have their own perceptions of what 'structurally different' means in the definitions above. Work is currently in progress to remove this element of subjectivity by folding the selection of the transmissibility regions into the overall optimisation problem.

In total, 44 features were assessed as strong or fair. For each of these features, a novelty detector was constructed using outlier analysis (see Appendix A). Each detector gave a scalar variable or novelty index which could be tested against a threshold, with exceedance signalling damage. Fig. 4 shows a novelty index over all of the damage states that was selected for its performance in identifying the removal of P1. The horizontal dashed line in the figure is the threshold for 99% confidence in identifying an outlier. The x-axis labels the damage state and there are three repetitions as the data were divided into a training set, validation set and testing set for neural network training. (As there were two separate panel removals for each damage state and 100 separate transmissibilities were measured each time, there are 66 points in the training, validation and testing sets).

The final stage of the analysis was to produce a damage location system. The algorithm chosen was a standard Multi-Layer Perceptron (MLP) neural network [10]. The idea was simply to map a set of the novelty indices obtained from the transmissibilities to the damage location, or in this case, to which panel was removed. An arbitrary decision was made to select 9 novelty indices, each of which was effective for one panel removal. The selection process is described in Ref. [4]. Although the selection process was informed by statistical evidence, it can only be termed *semi-objective* as there was no objective formula for trading between false-positives and false-negatives for the novelty indices, the final selection was based on an overall assessment of the misclassification rate associated with each novelty index. The neural network was supplied with the values of the 9 novelty indices at the input layer and required to predict the damage class at the output layer, more details can be found in Appendix B.

The 44 strong/fair features were given an integer identifier in the range 1–44. The semi-objective feature selection procedure gave the following feature set for neural network training: 2:3:5:7:14:X:28:33:43. For

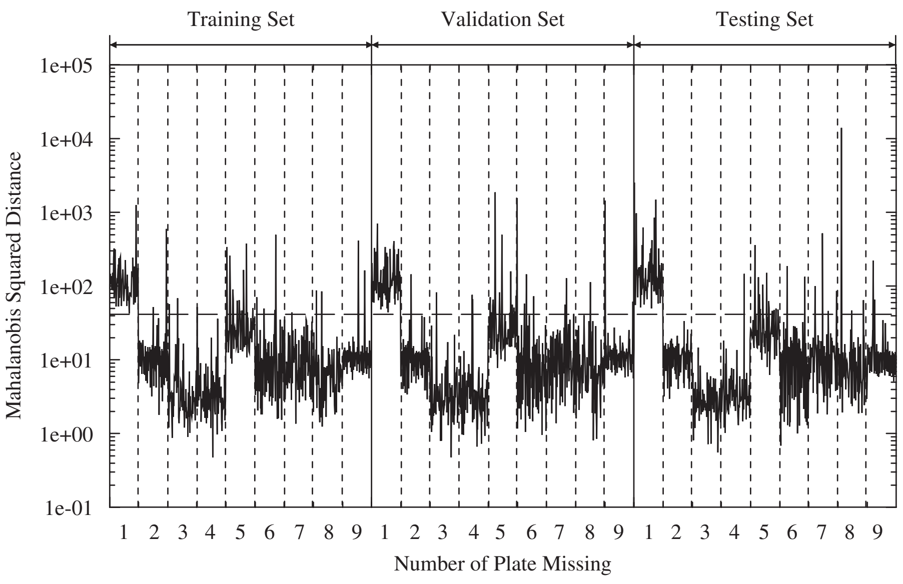

Fig. 4. Novelty index for various damages.

example, feature 2 is from transmissibility reciprocal T2*, lines 667 to 676 (10-dimensional), feature 7 is from T4*, lines 249 to 258 (again 10-dimensional). The feature denoted X is a weak feature which was originally selected on a visual basis as the best vector for distinguishing the P6 removal. There were no fair or strong features deemed suitable for this purpose.

The network training procedure followed the guidelines in Ref. [10]. The data was divided into a training set, a validation set and a testing set. For the network structure, the input layer necessarily had 9 nodes, one for each novelty index, and the output layer had 9 nodes, one for each class (panel). The number of hidden nodes was determined during training. In order to estimate the classification accuracy of each network the 1 *of M* strategy was used. This means that each class was assigned a specific network output [10]. After using this strategy, if the network is presented with a test vector, one assigns the appropriate class by finding the highest output.

At the end of this procedure, the network has in a sense been tuned to both the training and validation sets and therefore an independent testing set was required for proper verification of the network.

When the networks were trained, the one that gave the lowest validation error had 10 hidden units and gave a misclassification error of 0.155, corresponding to a training error of 0.158. The best results were obtained after 150,000 presentations of the data. The network weights were updated after each presentation, i.e. the training epoch was 1. When the best network was tested, it gave a generalisation error of 0.135, i.e. 86.5% of the patterns were classified correctly. As one might expect, the most errors were associated with panels 3 and 6 (the smallest). There was also some confusion between panel 8 and 9 removals, all of this is evident from the confusion matrix in Table 2.

Although the results were excellent, there was certainly room for improvement and it was postulated that an advance could be made by using a more principled optimisation scheme for the feature-selection process.

4. Genetic feature selection

For a more detailed analysis of GAs, the reader is referred to Ref. [11]. Very briefly, GAs are search procedures based on the mechanism of natural selection. A *population* of solutions is iterated, with the fittest solutions propagating their genetic material into the next generation by combination with other solutions. In the simplest form of GA, each possible solution is coded into a binary bit-string which constitutes the individual. Mating is implemented by exchanging corresponding sections of pairs of individuals. Mutation is also simulated by the occasional random switching of a bit.

Table 2
Confusion matrix for best neural network using semi-objective method of feature selection — testing set

Prediction	1	2	3	4	5	6	7	8	9
True class 1	62	1	0	0	2	0	0	1	0
True class 2	0	61	0	0	5	0	0	0	0
True class 3	0	1	52	0	7	4	0	2	0
True class 4	1	0	3	60	0	1	0	1	0
True class 5	2	1	0	0	60	3	0	0	0
True class 6	2	0	6	0	8	52	0	0	0
True class 7	1	0	4	0	1	1	58	1	0
True class 8	0	0	0	0	1	1	0	62	2
True class 9	2	1	1	0	0	0	0	15	47

It was shown in Ref. [6], that the binary representation is unsuitable for the sensor optimisation problem. As a result a modified GA is used, where the gene is a vector of integers, each specifying a feature; i.e. the individual 2:14:17 represents a 3-feature distribution, with each integer a feature index (in the range 1–44 here). The operations of reproduction, crossover and mutation for such a GA are straightforward modifications of those for a binary GA [11].

The initial population for the GA was generated randomly as standard.

The individuals — in this case feature distributions — are propagated according to their fitness. Here, this was evaluated as follows: for each feature distribution, a neural network diagnostic was trained and the probability of error was evaluated on the validation set. The inverse of this quantity was used as individual fitness. The training data for each network were generated by restricting the full 44-feature candidate patterns to the subset specified by the individual. In the GA runs, distributions with repeated features were penalised.

The parameters used for the GA runs were as follows: population size of 50, number of generations 100, probability of crossover 0.75, probability of mutation 0.05. A single member elite was used and five new blood were added at each iteration. Single-point crossover was used. For explanations of all these terms, refer to Ref. [11]. Because the algorithm is stochastic by nature, five runs of the GA were made and the best fitness taken to indicate the solution. It was decided to optimise the feature selection and the network structure separately, so the number of hidden units in the neural network was set at 10 and the training was conducted for a fixed number of 100,000 iterations (with an epoch of 1).

After five runs of the GA, the best fitness obtained on the validation set was 21.2143, corresponding to a probability of error of 0.047. The classification rate was therefore 95.29%, a marked improvement on the semi-objective feature selection used in Ref. [4]. However, note that this is on the validation set. One might argue that this is a valid measure of network generalisation as the network structure and training time were fixed. However, this is not the case as the network has been tuned to the given feature set and should still be assessed on an independent data set. The best 9-feature distribution was 2:3:4:7:22:23:24:34:42, i.e. the optimisation procedure chose only three of the features (2, 3 and 7) chosen by the semi-objective method.

The network structure (starting conditions, number of hidden units and stopping time) were then optimised according to Tarassenko's scheme discussed earlier ([10]) using the best 9-feature feature set above. The best network had 24 hidden units and was trained for 120,000 cycles. The training error was 0.044, the validation error was 0.040 and the final testing error was 0.066. This corresponds to a classification rate of 93.4%, an improvement of 6.9% on the semi-objective method. The confusion matrix on the testing set is listed in Table 3.

The confusion matrix shows that not only has the overall performance of the locator been improved; the panel 3 removals no longer prove problematical and the ambiguity between the removals of panels 8 and 9 has been eliminated. The performance relating to the panel 6 removals could still be improved.

Before testing was carried out, it was observed that the novelty indices had very large ranges (note that Fig. 4 is plotted on a log scale), and that the input scaling of the neural network onto the interval $[-1, 1]$ might well be weighting out lower values of the indices with potentially useful information content. It was therefore decided to transform the initial features by taking the natural logarithm. Clearly, there was no need to change

Table 3
Confusion matrix for best neural network using linear features — testing set

Prediction	1	2	3	4	5	6	7	8	9
True class 1	62	1	1	0	1	0	0	0	1
True class 2	0	62	4	0	0	0	0	0	0
True class 3	0	0	64	0	0	1	0	0	1
True class 4	1	0	0	62	0	3	0	0	0
True class 5	1	1	1	0	59	1	3	0	0
True class 6	2	3	0	0	0	58	2	1	0
True class 7	1	0	0	0	0	0	64	0	1
True class 8	0	0	0	0	3	0	0	62	1
True class 9	1	1	1	0	0	1	0	0	62

Table 4
Confusion matrix for best neural network using log features — testing set

Prediction	1	2	3	4	5	6	7	8	9
True class 1	65	0	0	0	0	0	0	0	1
True class 2	0	65	0	1	0	0	0	0	0
True class 3	1	0	62	0	0	1	0	1	1
True class 4	0	0	0	66	0	0	0	0	0
True class 5	0	0	0	0	66	0	0	0	0
True class 6	0	3	0	0	0	62	0	1	0
True class 7	0	0	0	0	0	0	66	0	0
True class 8	1	0	0	0	0	0	0	65	0
True class 9	0	0	0	0	0	0	0	0	66

the indexing. Five runs of the GA were made with the same GA parameters as before and the same neural network parameters. This time, the GA found the same maximum fitness of 99.0001 in two out of the five runs. This corresponds to a probability of error of 0.0101 on the validation set or a classification rate of 98.99%. The two fittest feature distributions were: 1:2:3:7:9:28:31:33:44 and 1:2:3:6:15:25:32:33:34. The first agrees with the semi-objective method on 5 features and the second on 3 features. For the purposes of brevity, only the first of these distributions will be discussed in any detail.

The network structure (starting conditions, number of hidden units and stopping time) were then optimised according to Tarassenko's scheme discussed earlier using the first of the two 9-feature sets above. The best network had 10 hidden units and was trained for 80,000 cycles. The training error was 0.0034, the validation error was 0.0084 and the final testing error was 0.0185. This corresponds to a classification rate of 98.1%, an improvement of 11.6% on the semi-objective method. As an alternate measure of performance, the network made only 11 misclassifications on the testing set of 594 patterns. The confusion matrix on the testing set is listed in Table 4.

Fig. 5 shows the evolution of the fitness throughout the optimisation for the first log feature distribution using 9 features given above. The GA converged on a good solution in 22 generations, corresponding to 1100 function evaluations. This is an excellent performance given that the set of possible distributions has in excess of 10^8 elements. As each neural network takes approximately 4 s to train, the optimum was arrived at in about 73 min, (it may be noted that five runs were carried out, so the overall time to arrive at the solution would be five times this).

The first optimal 9-feature distribution used transmissibilities measured across panels: 1, 2 (twice), 4, 5, 7 (twice), 8 and 9. This is very interesting and supports the idea used in Ref. [4] that one should maximise geometrical coverage of the wing. The only transmissibilities omitted are those measured over the small panels 3 and 6. This is a little surprising as one might expect them to yield vital information. However, it appears that there is enough information in the other features to adequately diagnose removal of the small panels.

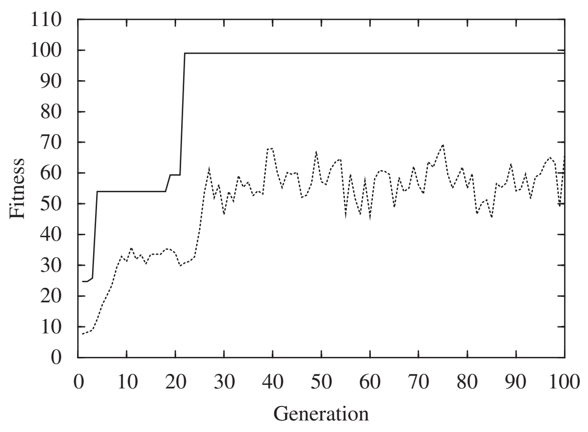

Fig. 5. Evolution of fitness function for optimisation of 9 features. Maximum fitness (solid line), average fitness (dotted line).

Table 5
Best sensor distributions using linear features

Number of features	Feature distribution
1	34
2	4:34
3	4:34:35
4	3:4:34:40
5	2:3:4:24:34
6	2:3:4:27:33:34
	2:3:4:7:24:34
7	2:3:4:7:23:28:34
8	2:3:4:6:27:34:42:44
9	2:3:4:7:22:23:24:34:42
10	2:3:4:6:9:23:24:34:42:44
	2:3:4:6:12:24:25:34:35:42

The improvement in classifier performance with a more principled feature-selection strategy allows for two possibilities. Firstly, one can maintain the use of 9 features for the classifier, in which case one obtains a marked improvement in performance. A second possibility is to accept a lower level of performance with a reduced feature set. Although the second option would not be appropriate for a safety-critical problem like damage identification, it is interesting to see how smaller feature sets, chosen on the basis of optimisation, can perform. In order to investigate this, the GA was allowed to optimise feature sets containing between 1 and 10 features. In each case, 5 runs of the GA were made with different initial populations. The resulting optimal distributions are listed in Tables 5 and 6 for the linear and log features respectively.

The selection of features indicated in Tables 5 and 6 is summarised in Figs. 6 and 7 which show frequency histograms for selection of the linear and log features respectively. Two points are apparent. The first is that the two histograms agree on the most important features (2,3 and 34) as one would hope. The second point is that there is a wider selection of log features chosen than the linear ones. Again, this is to be expected as the operation of taking logs was intended to reveal information at low levels of novelty which were not visible in the linear features.

Once the optimal distribution was found for each feature number, the neural network structure was optimised for number of hidden units, initial conditions and stopping time following best practice as highlighted by Tarassenko in Ref. [10]. The resulting error rates for each number of features are listed in Table 7 together with the number of hidden units selected with the consequent number of network weights.

Table 6
Best sensor distributions using log features

Number of feature	Feature distribution
1	43
2	23:44
3	3:15:35
4	3:4:23:39
5	2:4:7:34:35
6	2:3:6:15:28:35
7	2:3:7:15:28:34:39
8	1:2:3:7:9:25:34:43
	2:3:4:7:15:26:27:35
	2:3:4:6:15:33:34:35
9	1:2:3:6:15:28:32:33:34
	1:2:3:7:9:28:31:33:44
10	1:2:3:6:9:23:25:26:34:39
	1:2:3:4:7:9:28:33:34:39
	2:3:4:7:15:18:28:29:34:39

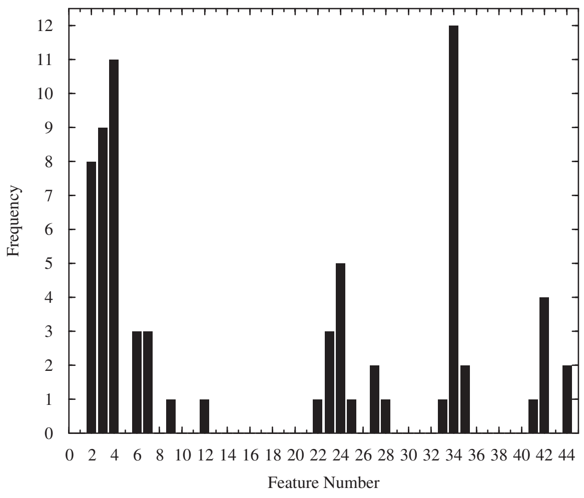

Fig. 6. Number of times features were selected during genetic optimisation: linear features.

Table 7
Best neural networks for each feature number: linear and log features

Number of features	Linear features			Log features		
	Error rate	Number of units	Number of weights	Error rate	Number of units	Number of weights
1	0.700	2	31	0.549	27	306
2	0.525	4	57	0.357	6	81
3	0.337	30	399	0.189	26	347
4	0.283	20	289	0.118	19	275
5	0.163	30	459	0.066	28	429
6	0.146	24	393	0.054	16	265
7	0.091	29	502	0.022	17	298
8	0.074	25	459	0.020	14	261
9	0.066	24	465	0.019	10	199
10	0.060	14	289	0.020	10	209

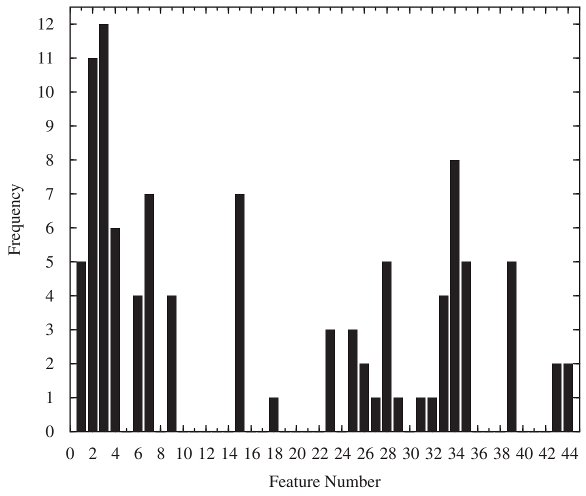

Fig. 7. Number of times features were selected during genetic optimisation: log features.

Noting that the error rate corresponding to the semi-objective method of choosing 9 features was 0.135 (on the testing set), one can see that a reduction to 7 linear features or 4 log features is possible with no reduction in performance, when the features are optimised in a principled manner.

5. Issues of network generalisation

The results listed in Table 7 indicate that neural networks with quite a large number of weights were necessary to give the required performance. This would normally raise questions about the generalisation capacity of the resulting networks, given that there is a comparatively small amount of training data, 594 patterns to be specific. An often-quoted rule of thumb, Ref. [10] is that, to avoid over-training of neural networks, one should have 10 training patterns for each weight in the network. This is clearly not the case here. A more analytical guide, based on Vapnik's statistical learning theory [12], is given in Ref. [13]. It is stated that, to correctly classify a fraction $1-\varepsilon$ of new patterns, the training set should contain a number of patterns n given by at least,

$$n = \frac{W}{\varepsilon}, \tag{1}$$

where W is the number of weights in the network. For an error rate of $\varepsilon = 0.1$ or a 90% success rate, this agrees with the ten-pattern rule discussed above. In order to achieve the success rate indicated by the semi-objective approach, one would require a minimum number of patterns given by,

$$n = 7.41W \tag{2}$$

whereas, the ratio actually found for the optimum 9 log-feature network is $n = 2.98W$. In other cases for different numbers of features the ratio is worse than this particularly for the linear feature sets. There are two comments which can be made in defence of the analysis here and one to the contrary.

In defence of the analysis, one might first say that the theory that leads to equation (1) actually guarantees only *worst-case bounds* on generalisation performance and it would be hoped in practice that one could achieve good generalisation with fewer training patterns than suggested by Eq. (1). The second comment in support of the results here is that the neural network training has actually been carried out in a principled manner which incorporates the practice of 'early stopping' i.e. networks are chosen which have been trained to the point where the validation error reaches its minimum and begins to rise again. Such an approach has been shown to be equivalent to weight-decay regularisation [14] and is an aid to generalisation.

The problem with the analysis here relates to the second observation above. Early-stopping relies on the existence of a validation set which is appropriately 'independent' of the training set. If the training and validation sets are too highly correlated, then the validation error will decrease as long as the training error decreases and this will invalidate the practice of early-stopping. In fact, the independence of the training and validation sets used here could be called into question. The reason for this is that the training, validation and testing sets were obtained by taking all the measured patterns for the fault cases and then cycling through this set taking every third pattern into the training data and so on. One might argue that this would maximise the correlations between the training and validation sets. This strategy was used here for a good reason. When the damage cases were examined, 100 patterns were recorded for each panel removed and then later another 100 patterns were recorded for a completely independent removal of the same panel. One might argue that 66 of the first 100 points should have been used for training while 66 of the second 100 points should be reserved for validation. In fact this approach would have been doomed to failure because of the large operational variations in the data as a result of fixing the panels down each time with different boundary conditions. The work in Ref. [3] showed clearly that the effects of the boundary conditions are able to conceal the effects of the faults. As there was not enough time in the experimental program to consider many panel removals and thus sample the whole range of operational conditions, it was decided to interleave the training, validation and testing sets as described above, in order to make the most of the available data. Further evidence that the neural networks above with greater numbers of hidden units might be overtrained was suggested by carrying out a thresholding analysis as discussed in Ref. [15]; however, it is beyond the scope of this paper to discuss this technique in any detail.

Because there is evidence both for and against overtraining here, it was decided to investigate the effect on network performance of reducing the allowed number of hidden neurons. In the exercises described earlier, the neural networks were allowed to have 10 different initial conditions, stopping times up to 200,000 presentations of data and up to 30 hidden units. The exercise was repeated, using the linear features as they generally produced the networks with the most weights, but only allowing up to 10 hidden neurons. The results obtained are listed in Table 8.

Note that the results in Table 8 are the *testing* errors, so it is not impossible for lower errors to be obtained than in the earlier simulations. As one can see from Table 8, there is no marked deterioration in the results when the number of hidden units are restricted. In fact, the overall best error is obtained for a 10-feature network and is lower than that previously obtained with more hidden units (Table 7). The best results are obtained with a network with 6 hidden units, for which $n = 4.6\ W$. In the case of the networks trained with log features, the best performance was actually obtained with a network with 10 hidden units, so a similar exercise would be unnecessary.

A further exercise was carried out with a restriction to a maximum of 5 hidden units and the results are listed in Table 9. As one might expect, this does lead to a marked deterioration in performance, although the best error rate obtained (0.079 at 10 features) is still much better than that obtained with the semi-objective

Table 8
Best neural networks for each feature number: linear features and a maximum of 10 hidden units

Number of features	Linear Features		
	Error rate	Number of units	Number of weights
1	0.700	2	31
2	0.525	4	57
3	0.340	10	139
4	0.279	7	107
5	0.180	8	129
6	0.124	10	169
7	0.091	7	128
8	0.074	9	171
9	0.072	9	180
10	0.057	6	129

Table 9
Best neural networks for each feature number: linear features and a maximum of 5 hidden units

Number of features	Linear features		
	Error rate	Number of units	Number of weights
1	0.700	2	31
2	0.525	4	57
3	0.337	4	61
4	0.278	4	65
5	0.187	5	84
6	0.170	4	73
7	0.099	5	94
8	0.104	5	99
9	0.104	5	104
10	0.079	5	109

approach to feature selection. The log features fare better here and the 9-feature network with 5 hidden nodes achieves an error rate of 0.04 corresponding to a 96% success rate which is radically better than that achieved using the semi-objective approach. The 9-feature log network has $n = 5.7\ W$.

In summary, even taking into account concerns about the generalisation capacity of some of the networks in Table 7, it is clear that superior results are obtained using the optimisation procedure even with marked restrictions on the number of hidden units. In the case of the 9 log-feature networks, the performance only falls from 98.1% to 96% when the number of hidden neurons is reduced from 10 to 5. Note that low model complexity is not a guarantee in itself of generalisation capability; however, it is at least a necessary condition.

6. Discussion and conclusions

The first conclusion is rather simple and almost to be expected. In the recent past, the genetic algorithm has proved itself to be extremely powerful at solving combinatorial optimisation problems in general and sensor/feature optimisation problems in particular. It is no surprise that it proves advantageous in the SHM context described in this paper. Together with a simple feature transformation (taking logarithms), the GA feature-selection algorithm provides a 9-feature distribution with a classification rate of 98.1% (on testing data) compared with a rate of 86.5% based on a semi-objective selection strategy and linear features. Given that the optimisation procedure is reasonably inexpensive (computationally), there is every reason to recommend its use for general feature-selection problems. There is still work to be done in removing the element of subjectivity present in the current approach to feature selection. This can be addressed by folding the selection of the transmissibility frequency ranges into the overall optimisation procedure. This is a more difficult task and work is in progress. A further step in the direction of complete automation of the feature-selection/diagnosis process would be to also include the optimisation of the sensor positions. This is even more difficult and an effective strategy will require considerable further research.

The second main conclusion is not directly concerned with the optimisation but has been forced by the re-evaluation of the data and is more a matter of generalisation of data-driven models like neural networks. It is an undisputed fact that the study here has a small amount of data available for training. This is ultimately due to the fact that the aircraft concerned was only available in a one week window. This meant that despite the fact that the boundary conditions of the panel were known to be a matter of some concern, only two repetitions of each panel removal were possible. The way that the training, validation and testing sets were constructed here, means that the neural networks were able to distinguish different fault cases, despite the fact that each case spanned two boundary conditions. However, experience with the problem suggests that two samples is not enough to span the range of possible operational conditions in this case and that the neural networks produced may not generalise to unseen data from different fixing conditions. One is forced to the conclusion that if one is to pursue data-driven approaches to damage identification, one must take the time to acquire appropriate data spanning all possibilities. It may therefore be the case that the expense of the acquisition process rules out data-driven approaches in some circumstances.

Acknowledgements

The main acknowledgement must be to DERA (now QinetiQ), Air Systems Sector, Structures Department for funding the earlier stages of this project as part of the DTI CARAD scheme. The authors are grateful to the late Dr. David Allman for his invaluable collaboration on the overall SHM programme and to Mrs. Sue Copley for her patience and help in administering the project. Dr. Graham Skingle gave valuable technical support during the testing phase at Farnborough.

Appendix A. Outlier analysis

A.1. Outliers in univariate data

A discordant observation or outlier in a data set is an observation that is surprisingly different from the rest of the data in some sense, and therefore is believed to be generated by a different mechanism to the other data. The discordancy of a candidate outlier is some measure that can be compared against some corresponding objective criterion, and allows the outlier to be judged as statistically likely or unlikely to have come from the assumed generating model. The application to damage detection is clear; the discordancy should be evaluated with respect to a model constructed from a normal condition of the system of interest. The standard reference for outlier analysis is [16].

The case of outlier detection in univariate data is relatively straightforward and there are numerous discordancy tests. One of the most common, and the one whose extension to multivariate data will be employed later, is based on deviation statistics and given by,

$$z = \frac{|x_\zeta - \overline{x}|}{\sigma_x},$$

where x_ζ is the potential outlier and $\overline{x}$ and σ_x are the mean and standard deviation of the undamaged sample respectively. This discordancy value is then compared to a threshold value and the observation declared, or not, to be an outlier. The value of the threshold is critical; unfortunately it is usually necessary to make some assumptions about the data in order to establish it. Suppose the normal condition data are assumed Gaussian. In this case, there is a 95% probability that a sample drawn from the same distribution will lie in the range $[\overline{x} - 1.96\sigma_x, \overline{x} + 1.96\sigma_x]$. If a point observed during the monitoring period lies outside this range, there is only a 5% chance that the point is a sample from the same normal condition distribution. In practice the reference set will not be Gaussian, however if the deviation is small, the assumption of Gaussianity may yield a sensible threshold.

A.2. Outliers in multivariate data

A multivariate data set consisting of n observations in p variables may be represented as n points in a p-dimensional space.

The discordancy test that is the multivariate equivalent of the equation above is the Mahalanobis-squared distance measure given by,

$$D_\zeta^2 = (\underline{x}_\zeta - \underline{\overline{x}})^T [S]^{-1} (\underline{x}_\zeta - \underline{\overline{x}}),$$

where $\underline{x}_\zeta$ is the potential outlier, $\underline{\overline{x}}$ is the mean of the sample observations and $[S]$ the sample covariance matrix.

As in the univariate case, the Mahalanobis-squared distance of the potential outlier is checked against a threshold value, as in the univariate case, and its status determined. In the multivariate case, the threshold depends on the number of observations in the training set and the dimension of the feature vector. In this paper it is computed as defined in Ref. [17].

Appendix B. Neural network algorithm for localisation

The damage location system was based on a neural network. The specific algorithm chosen here was a standard multi-layer perceptron (MLP) neural network [13]. The idea was simply to map the individual novelty indices obtained from the transmissibilities (by outlier analysis) to the damage location, or in this case, to which panel was removed. The neural network was supplied with the values of the 9 novelty indices (one associated with each panel) at the input layer and required to predict the damage class at the output layer.

Note that there are now two layers of feature extraction. At the first level, certain ranges of the transmissibilities were selected for sensitivity to the various damage classes. These were used to construct novelty detectors for the classes. At the second level of extraction, the 9 indices themselves are used as features for the damage localisation problem. This depends critically on the fact that the various damage detectors are *local* in some sense, i.e. they do not *all* fire over *all* damage classes. For some assurance of this point, the reader is referred to Ref. [4]. The overall approach is summarised in Fig. 8, note that there may be repetition of actual physical sensors in the list $1, \ldots, N$ for the generation of features.

The procedure followed the guidelines in Ref. [10]. The data was divided into a training set, a validation set and a testing set. For the network structure, the input layer necessarily had 9 nodes, one for each novelty index, and the output layer had 9 nodes, one for each class. The number of hidden nodes was determined during training. In order to estimate the classification accuracy of each network the 1 *of M* strategy was used. This means that each class was assigned a specific network output. During training the network was required to respond with unity at the output corresponding to the presented class and zero elsewhere. The rationale for this training scheme is as follows. It can be shown from Ref. [13], that after training with the 1 *of M* strategy, if the network is presented with a test vector it will respond at the outputs with the Bayesian *a posteriori* probability that the input vector belongs to that class. In order to assign the class then, one simply finds the highest output, i.e. the class with highest posterior probability. There are a number of conditions which must apply for this to work correctly, one is that the prior probabilities of each class must be specified. In this case they were assumed equal, i.e. all damage types are equally likely. In order to enforce the condition of equal priors, a balanced training set was used with equal numbers of examples from each class, 66 from each.

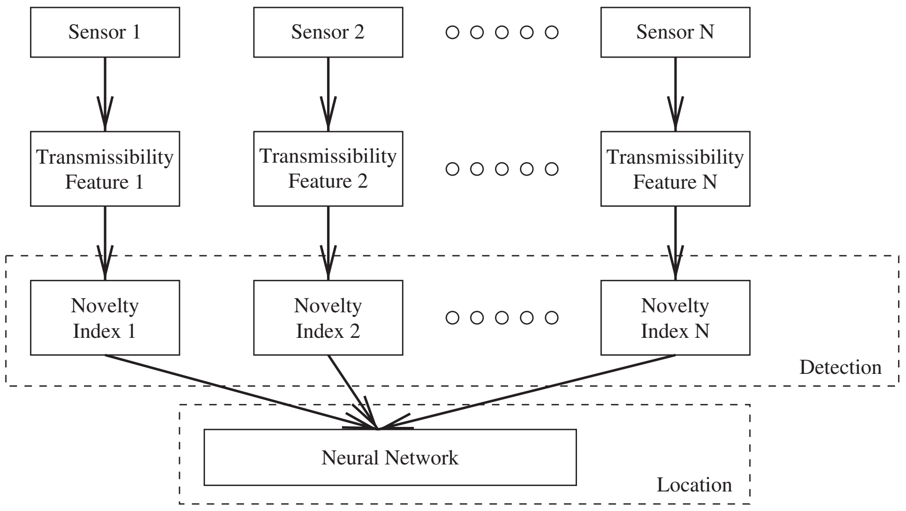

Fig. 8. Schematic showing damage location algorithm.

In terms of pseudo-code, the training strategy was:

```
for number of hidden layer neurons = 1 to 30
              {
              for different random initial conditions = 1 to 10
                {
                train network on training data
                evaluate on validation data
                terminate training at minimum in validation set error
                }
              }
```

This means that the training set is used to establish the weights. The structure and training time, etc. are optimised by selecting the conditions which give the lowest validation set error. At the end of this procedure, the network has in a sense been tuned to both the training and validation sets and therefore an independent testing set is required for proper verification of the network.

References

[1] A. Rytter, Vibration-based Inspection of Civil Engineering Structures, PhD Thesis, Department of Building Technology and Structural Engineering, University of Aalborg, Denmark, 1993.

[2] K. Worden, G. Manson, D.J. Allman, Experimental validation of a structural health monitoring methodology, part I: novelty detection on a laboratory structure, *Journal of Sound and Vibration* 269 (2003) 323–343.

[3] G. Manson, K. Worden, D.J. Allman, Experimental validation of a structural health monitoring methodology, part II: novelty detection on an aircraft wing, *Journal of Sound and Vibration* 269 (2003) 345–363.

[4] G. Manson, K. Worden, D.J. Allman, Experimental validation of a structural health monitoring methodology, part III: damage location on an aircraft wing, *Journal of Sound and Vibration* 269 (2003) 365–385.

[5] G. Manson, K. Worden, D.J. Allman, Experimental validation of a damage severity method, *Proceedings of First European Workshop on Structural Health Monitoring*, Paris, France, 2002, pp. 845–852.

[6] K. Worden, A.P. Burrows, G.R. Tomlinson, A combined neural and genetic approach to sensor placement, *Proceedings of the 13th International Modal Analysis Conference*, Nashville, USA, 1995, pp. 1727–1736.

[7] K. Worden, A.P. Burrows, Optimal sensor placement for fault diagnosis, *Engineering Structures* 23 (2001) 885–901.

[8] L.B. Jack, A. Nandi, Genetic algorithms for feature selection in machine condition monitoring with vibration Signals, *IEE Proceedings—Vision Image and Signal Processing* 147 (2000) 205–212.

[9] H. Sohn, Effects of environmental and operational variability on structural health monitoring, *Transactions of the Royal Society, Series A* 365 (2007) 539–560.

[10] L. Tarassenko, *A Guide to Neural Computing Applications*, Arnold, London, 1998.

[11] D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, MA, 1989.

[12] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer, New York, 2001.

[13] C.M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, Oxford, 1998.

[14] C.M. Bishop, Training with noise is equivalent to Tikhonov regularisation, *Neural Computation* 7 (1995) 108–116.

[15] S.G. Pierce, K. Worden, G. Manson, Evaluation of neural network performance and generalisation using thresholding functions, *Neural Computing and Applications*, 2007, to appear.

[16] T. Barnett, T. Lewis, *Outliers in Statistical Data*, third ed., Wiley, Chichester, 1994.

[17] K. Worden, G. Manson, N.R.J. Fieller, Damage detection using outlier analysis, *Journal of Sound and Vibration* 229 (2000) 647–667.