Journal of Sound and Vibration (2000) **238**(3), 401–424
doi:10.1006/jsvi.2000.3105, available online at http://www.idealibrary.com on IDEAL®

MEASURING AND IMPROVING NEURAL NETWORK GENERALIZATION FOR MODEL UPDATING

R. I. Levin, N. A. J. Lieven and M. H. Lowenberg

*Department of Aerospace Engineering, University of Bristol, Bristol BS*8 1*TR, England.*
E-mail: aero-office@bristol.ac.uk

(*Received* 6 *January* 2000, *and in final form* 8 *May* 2000)

This paper compares various techniques of measuring the generalization ability of a neural network used for model-updating purposes. An appropriate metric for measuring generalization ability is suggested, and it is used to investigate and compare various neural network architectures and training algorithms. The effect of noise on generalization ability is considered, and it is shown that the form of the noise does not appear important to the networks. This implies that the optimum training location may be obtained by considering a simple noise model such as Gaussian noise. Various radial basis function neurons and training algorithms are considered. Significant improvements to generalization ability are noted by merging the holdout and training data sets before training the second layer of the network, after the network architecture has been decided. The Gaussian radial basis function is rejected as the radial basis function of choice, due to uncertainty regarding an appropriate value for the spread constant. It is noted that several alternative radial basis functions without spread constants, such as the thin-plate spline, give excellent results. Finally, the use of jitter and committees to improve the generalization ability of networks is considered. It is found that jitter makes neither improvement nor degrades the results. It is also found that a committee of networks performs better than any single network. A good method of generating committee members is to split the available data evenly into multiple random holdout and training data sets.

1. INTRODUCTION

Previous work [1] introduced a method of updating finite element (FE) models using neural networks. A key property of neural networks is the ability to generalize, that is, to produce sensible results from data that have not previously been seen. The aim of this paper is to perform a detailed investigation of the nature of the network generalization, and to suggest improvements to neural network training algorithms.

Section 2 will discuss network generalization in more detail, and section 3 will investigate and compare various measures of generalization ability. In section 4, different approaches to training networks will be compared using an appropriate measure, and in section 5 several techniques to improve generalization will be considered. The aim is to produce an approach to neural network training that will result in the best generalization ability for model updating purposes. Great computational effort is required to generate training data for neural networks, and it seems prudent to spend some effort to obtain the maximum benefit from it.

0022-460X/00/480401 + 24 $35.00/0

2. NATURE OF NETWORK GENERALIZATION

Neural networks learn the target values of training data input vectors during training. However, the usefulness of the networks stems from their ability to respond sensibly to input vectors that were not contained in the training data set; that is, the ability to generalize.

In order for a network to generalize successfully, several conditions must be met:

- the underlying function must be sufficiently smooth; a network can learn a function with a finite number of discontinuities, but not totally chaotic or random functions;
- the training data must provide enough information to learn the underlying function, i.e., generalization may fail if there are hidden variables affecting the training data that are not shown to the network, or if noise has swamped the available information;
- the network must have enough "freedom" to learn the training data. This is a key point that will be considered in this paper; and
- the network should only be asked to respond to inputs that lie within the region of the training data (interpolate). If it is asked to respond in areas that lie without the training data (extrapolate) then the results will often be invalid.

All the networks used in the paper are radial basis functions. These are self-organizing networks and as such have not pre-defined structure. The network grows to fit the problem by adding neurons until a specified error goal is reached. Figure 1 shows an RBF network successfully modelling a sinusoid when trained with some noisy observations from the underlying function. The response of the network is sensible when interpolating, but the extrapolation ability of the network is very poor. It is unreasonable to expect a non-parametric regression technique such as neural networks to extrapolate well.

The problems of overtraining and undertraining a neural network are now briefly discussed. Figure 2 shows the responses of two different networks to the same noisy

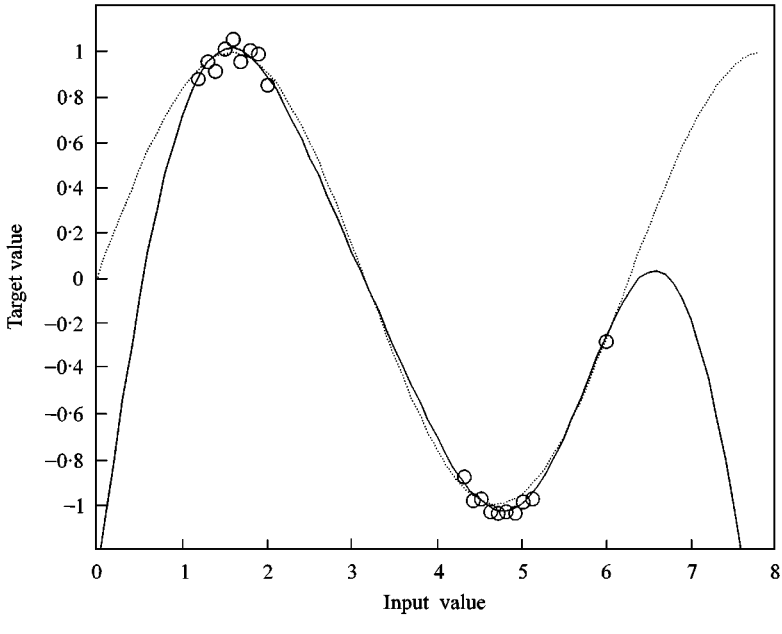

Figure 1. Interpolation versus extrapolation for a neutral network: ·····, underlying function; ——, network output; ○, training data point.

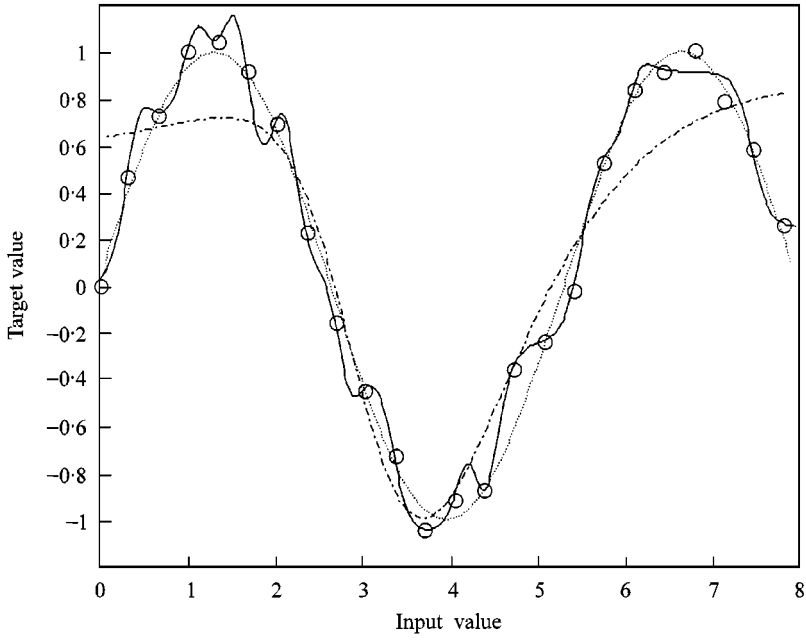

Figure 2. Underfitting and overfitting neural networks: ·····, underlying function; ——, overfitting output; – – – –, underfitting output; ○, training data point.

observations of the underlying function. The first network, shown as the dashed–dotted line, has clearly failed to model some of the main features of the underlying function; it has underfitted the data. The second network, shown as the solid line, has overfitted the data. This network has modelled the noise on the data, not the underlying function. Overfitting is problematic, as it can be difficult to distinguish between a network that has matched the training data well and one that has overfitted the training data.

The overfitting/underfitting dilemma (or trade-off) can also be considered in statistical terms as a bias/variance trade-off. Bias error refers to the difference between the underlying function and the average output of the neural network. For example, consider a network that is trained many times, each time with a different set of noisy data taken from the same underlying function. If the network consistently underfits the data, then it has a high bias error. The variance error refers to the sensitivity of the network outputs to the particular training data set chosen.

It can be seen that a network with few neurons will have a high bias error (i.e., fail to model the underlying function) but a low variance error (i.e., its response is not sensitive to the particular training data chosen). A network with a large number of neurons will tend to have a low bias but a high variance. Thus, there is a trade-off between bias and variance.

There is, however, one way to reduce both the bias and variance errors of a neural network model consistently, and that is to increase the amount of training data available [2]. Fortunately, for model updating it is possible to generate an amount of training data limited only by computer time.

The next section will consider how to measure the generalization ability of a neural network for model updating purposes. This measure will be used to find the optimum termination point during network training.

3. MEASURES OF GENERALIZATION

There are many possible strategies for measuring the generalization properties of neural networks. They can be divided into two broad categories: (1) holdout measures; and (2) whole set measures.

These categories of generalization measure will be compared using two-test FE models—a 10-element cantilevered beam and a free–free T-shaped plate.

3.1. FE MODELS AND UPDATING APPROACH

This section will describe the two simple test FE models, and the basic neural network updating approach used. See reference [1] for a more detailed explanation and an introduction to neural networks.

The 10-element cantilevered beam used was introduced in reference [1]. Twenty different updating parameters, or p-values, were used with this model, one for the mass and one for stiffness of each element. The choice of mass and stiffness as updating parameters is often inappropriate for large classes of model updating problems. The problem in this form is inherently ill-conditioned, which makes it the ideal example for the generalization simulations shown in this paper.

The second test structure used is a 10 element T-shaped plate (Figure 3) with free–free boundary conditions. The parameters used for each element of the plate are: density $= 7850\ \text{kg/m}^3$, Young's modulus $= 2{\cdot}1 \times 10^{11}\ \text{N/m}^2$, thickness $= 3{\cdot}5$ mm, the Poisson ratio $= 0{\cdot}3$. All elements are square, of dimension 100 mm $\times$ 100 mm.

Ten updating parameters are considered for the T-shaped plate, namely the 10 elemental stiffness parameters. Thus, the updating parameters p_1 to p_{10} correspond to the stiffness of elements 1–10 as labelled in Figure 3.

The updating procedure consists of generating a set of training data and using this data to train a Gaussian radial basis function (RBF) neural network, using the orthogonal least squares (OLS) training algorithm. The (simulated) measured experimental data are applied to the network, which then produces a set of FE model adjustments. This results in an updated FE model. Note that the spread constant of every RBF neuron in the first layer of the network is set equal to the mean of the Euclidean distance between all pairs of input vectors in the training data.

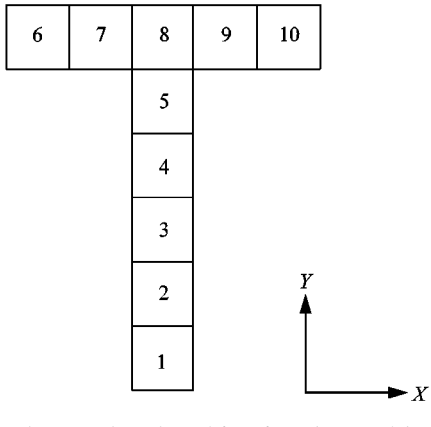

Figure 3. A T-shaped free–free plate model.

The training data consist of pairs of input and output vectors. Each output vector contains a particular set of alterations made to the p-values of the model. Each corresponding input vector contains a selection of the modal data resulting from applying these p-values to the model. Thus, the network should be able to learn how variations to the p-values affect the modal data.

For the cantilevered beam, the modal data used consists of the first three bending and two extensional modes, together with their natural frequencies. Rotational information was ignored, so each input vector contained five modes (of length 10) and five natural frequencies (scaled numerically to be of a similar order of magnitude to the mode shapes). Thus, the 20 p-values are estimated using input vectors of length 55.

For the T-shaped plate, only the out-of-plane translational d.o.f.s are included in the training data; in-plane translational and all rotational d.o.f.s are not retained. The model has 22 nodes, so each mode shape contributes 22 terms to the training data. The first five mode shapes and natural frequencies are shown to the network, so each experimental input vector is of length 115 (5 mode shapes of length 22, and 5 natural frequencies). All rigid-body modes are omitted from the training data. Note that examining the effects of including different amounts of modal data in the input vector on the updating procedure is outside the scope of this paper.

3.2. HOLDOUT MEASURES OF GENERALIZATION

With two test FE models and the basic updating procedure established, the remainder of section 3 will consider how to determine when to terminate the OLS training algorithm to produce an optimum network.

The generalization ability of an RBF neural network typically improves during the early stages of learning, as new neurons are added to the network. However, during later stages the network can overfit the training data, degrading its generalization ability. Holdout measures use a second set of data (the holdout or validation set) to score the generalization potential of the network at various stages of training. The holdout set is *not* used directly in training the network, but instead to determine the optimum point to stop training.

The optimum training termination point is typically chosen when the generalization error "starts to increase". However, since the generalization measurement can fluctuate, it can be difficult to decide whether an increase in generalization error is due to overfitting or a simple fluctuation. A robust solution is to measure the generalization error as each new neuron is added to the network, until all the training data are used. The network with the minimum generalization error is then used.

Before a holdout set can be used it is necessary to decide what data should make up the holdout set, and how the generalization ability of a given network should be measured with this set. This second question is not trivial; an inaccurate measure of generalization can skew the selection of the optimum network significantly.

3.3. EXAMPLES OF HOLDOUT DATA SETS

Two different types of training data were considered in reference [1]. The first, termed the *change-one* data set, consisted of varying each p-value in turn whilst holding the remaining p-values unchanged. This ensures that the network can see the effects of adjusting each p-value, but it does not include the effects of adjusting multiple p-values. Consequently, a second data set, the *line* data set was introduced. This involved setting a random selection

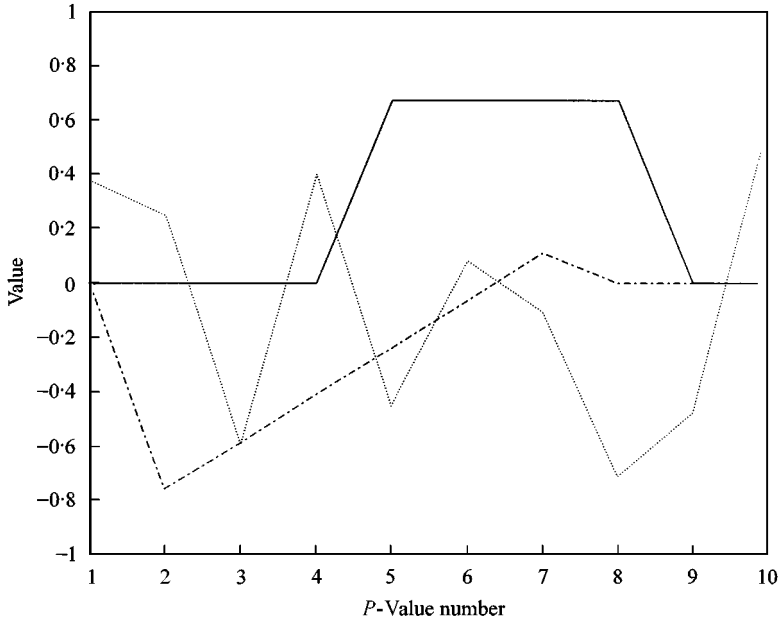

Figure 4. Typical target *p*-value vectors from the holdout data sets: ·····, rand holdout set; ——, line holdout set; – · – · –, diag holdout set.

of neighbouring *p*-values to a uniformly distributed random value. Figure 4 shows the *p*-values for a typical member of the *line* data set (solid line on graph).

The training data set consists of the union of the *change-one* and *line* data sets. For the *change-one* data set, each parameter is set in turn to $\{-0{\cdot}8, -0{\cdot}4, -0{\cdot}2, 0{\cdot}2, 0{\cdot}4, 0{\cdot}8\}$. 50 members of the *line* data set are added to this *change-one* set, with *line* *p*-values adjusted between $-0{\cdot}8$ and $0{\cdot}8$. Thus, the cantilevered beam training set contains 170 observations from the FE model, and the T-shaped plate training data set contains 110.

An additional holdout data set will be generated for both of the test cases to measure the generalization. However, it is not clear what form this holdout set should take; should it be additional observations from the *line* data set, an additional *change-one* observation, or a different type of data set that the network has not seen? Three different holdout data sets will be considered and their properties examined.

The first holdout set consists of 200 additional observations from the *line* data set. It is estimated that 200 observations will give an accurate measure of generalization ability, but this will be examined in more detail later. Note that the network has "seen" 50 observations from the *line* data set during training, so it is possible that the network response to this sort of example is unreasonably good and a different training set would produce worse results. Consequently, a second and third holdout set will be tested, using types of adjustments to the *p*-values that the network has not seen during training.

The second holdout set contains 200 observations from the *diag* set. Each member of the *line* set is generated by selecting random minimum and maximum *p*-value numbers, and setting all of the *p*-values between these numbers to a uniformly distributed random variable. For the *diag* set the selected *p*-values are instead set to linearly interpolated values between uniformly distributed independent start and end values. Consequently, the *diag* set is a superset of the *line* set. Figure 4 shows typical *p*-value settings for a member of the *diag* set (dash–dot line), for the T-shaped plate.

The third holdout set consists of 200 observations from the *rand* data set. *Rand* is the most general set possible; each *p*-value is independently, uniformly distributed between the extreme values. This data set is much more difficult for neural networks to model than the previous two sets. A typical member of the *rand* set is also shown in Figure 4 (dotted line). Note that for each of the three holdout sets the *p*-values are bounded between $-0{\cdot}8$ and $0{\cdot}8$, as for the training data.

3.4. GENERALIZATION ERROR METRICS

With the three holdout sets chosen, it remains to decide how to measure the generalization error of a network. A standard measure is the mean-squared error (MSE):

$$MSE = \frac{1}{N_{cases} N_p} \sum_{i=1}^{N_{cases}} \sum_{j=1}^{N_p} (\{y_i\}_j - f_j(\{x_i\}))^2, \tag{1}$$

where $\{x_i\}$ is the ith input vector, $\{y_i\}_j$ is the jth term of the ith target vector, f_j is the jth output from the network, N_{cases} is the number of cases in the training data set and N_p is the number of *p*-values (equal to the number of output neurons).

Figures 5 and 6 show how the MSE generalization measure varied for the three holdout sets during the training of both test cases. Several points are notable about these results. Firstly, the diagonal and line holdout data sets performed similarly for both of the test cases, but the random set performed very differently. These results actually show the failure of the networks to respond well to the random set, since similar MSE values result from randomly guessing the *p*-values instead of using the network response. This failure is perhaps not surprising. It would be an exceptional network that could generalize from the training data set to model the random set successfully.

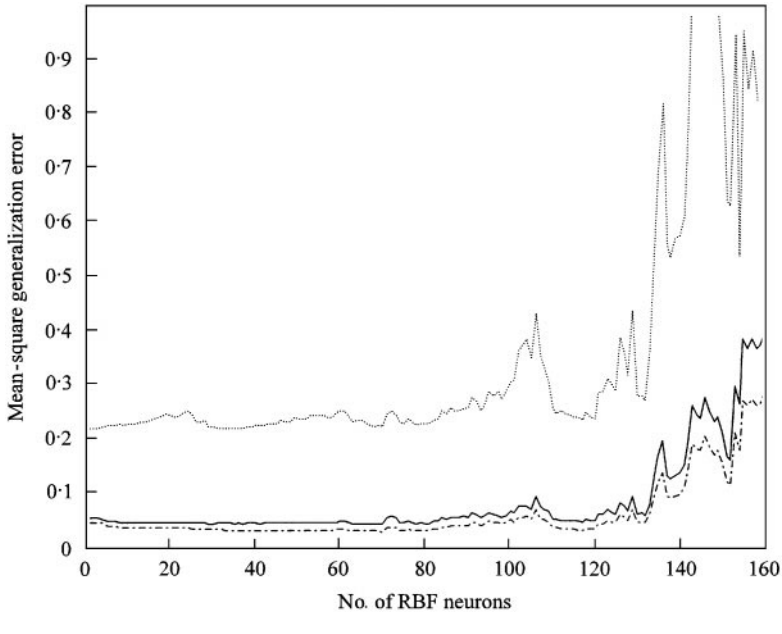

Figure 5. Generalization measure using holdout sets for the cantilevered beam: ·····, rand holdout set; ——, line holdout set; – – – -, diag holdout set.

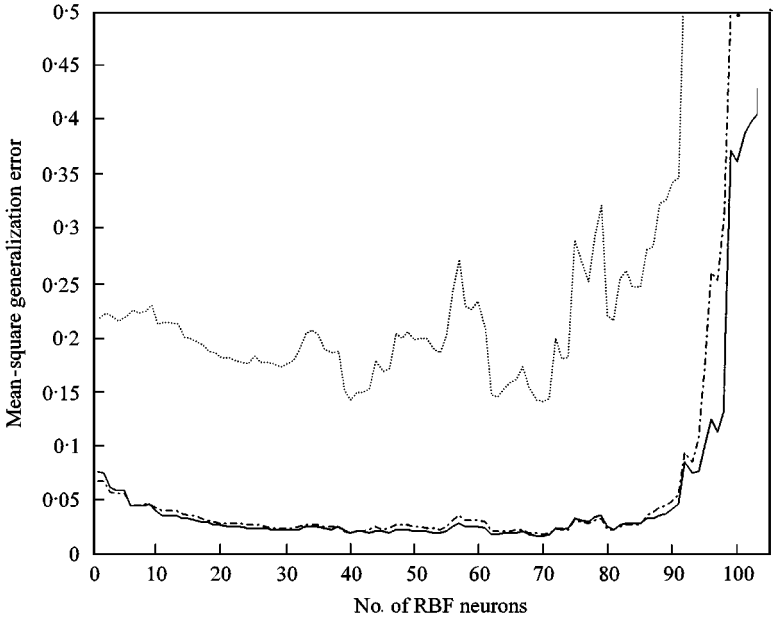

Figure 6. Generalization measure using holdout sets for the T-shaped plate: ·····, rand holdout set; ——, line holdout set; – – – -, diag holdout set.

The second point to note is that although the line and diagonal holdout sets give similar measures of the generalization error, the MSE curve is flat for a large range of neurons, especially for the cantilevered beam case. This causes concern, because the MSE results for a network containing one neuron are similar to those of a network containing 100 neurons. The network containing only one neuron will obviously underfit the underlying function, and should result in a poor generalization measure. Hence, selecting the network with the minimum MSE could result in selecting an inferior network.

The reasons for the poor performance of the MSE measure can be seen by examining the network response to a typical member of the holdout set, at various stages during training. Four difference network responses for a cantilevered beam holdout case are shown in Figure 7. The cantilevered beam has 20 *p*-values, but the *p*-values do not all have the same "strength". For example, alterations to the stiffness near the root (*p*-values, 1–4) or the mass near the tip (*p*-values, 17–20) will result in larger changes to the modal parameters than equivalent alterations made elsewhere on the model.

Column 3 of Table 1 shows the MSE results of the four cases considered in Figure 7. It is striking that the MSE value of case A (0·0376) is actually lower than that of case C (0·0468). Case C has produced good results for the least-sensitive *p*-values (1–6, 14–20) and poor results for the most sensitive *p*-values (7–13). The MSE has penalized this poor performance on the less important *p*-values.

This highlights a flaw in the MSE as a generalization measure for model updating purposes; a single erroneous *p*-value can result in a very poor score for an otherwise excellent result. This tends to flatter networks that are underfitting the underlying function. Clearly, a measure of generalization error is required that is both robust and capable of penalizing networks that are underfitting.

The generalization error measures can be made more robust by considering the absolute difference between the two *p*-value sets, instead of the squared difference. Additionally, the

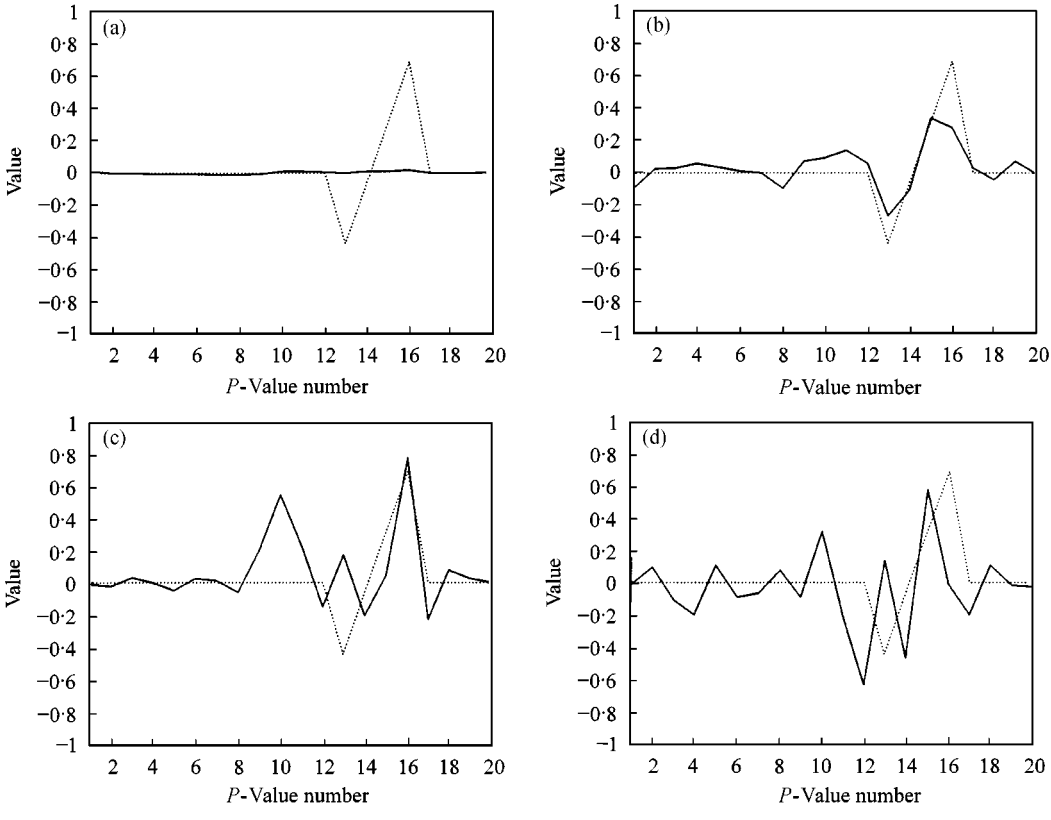

Figure 7. Various network responses to the cantilevered beam test case using a single holdout case: ·····, input value; ——, net response. (a) 1 neuron; (b) 55 neurons; (c) 75 neurons and (d) 150 neurons.

TABLE 1

The generalization measures for the four cases considered in Figure 7

Case	No. of neurons	*MSE*	Median absolute error	Median absolute error, non-zero terms only, (GENDIFF)
A	1	0·0376	0·0062	0·373
B	55	0·0134	0·0518	0·108
C	75	0·0468	0·0678	0·210
D	150	0·0860	0·1055	0·484

median can be used instead of the mean to compare each set of *p*-values. The use of the median tends to reduce the effects of a few erroneous *p*-values on the generalization error. *Note*: although the median is used to obtain a generalization error for each test case, the mean is used to average these individual errors over the whole test, because each test case is equally important. The results of using this median absolute error (MedAE in equation (2)) on the four test cases considered are shown in Table 1, Column 4:

$$\text{MedAE} = \frac{1}{N_{cases}} \sum_{i=1}^{N_{cases}} \underset{j=1,\,\ldots,\,N_p}{\text{median}} \left(\text{abs}(\{y_i\}_j - f_j(\{x_i\}))\right). \tag{2}$$

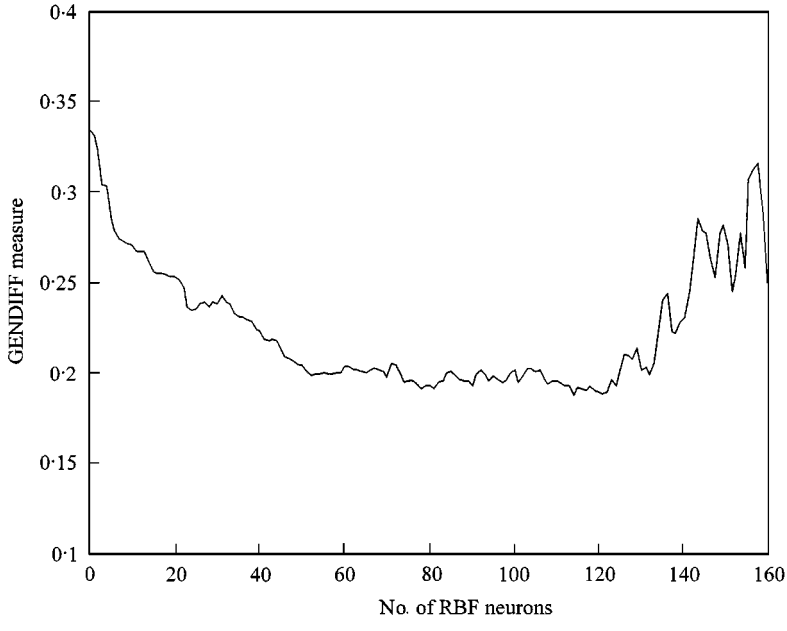

Figure 8. GENDIFF generalization measure using *diag* holdout set, cantilevered beam.

Again, these results do not reflect adequately the performance of the network. The underfitting network (case A) has been assigned an excellent generalization error of 0·0062. This has occurred because the median has eliminated the four erroneous non-zero p-values (nos. 13–16) from the measure, leaving only the 16 correct zero p-values. This new measure is not appropriate for holdout sets such as *diag*, where many of the target p-values are zero, because it flatters a network that simply returns zeros regardless of the input to the network.

The MedAE measure can be improved by considering only the p-values that have non-zero targets, and simply ignoring the p-values that have a zero target. The results obtained using this new measure, termed GENDIFF (see equation (3)), are shown in Table 1, Column 5. This new measure penalizes both networks that underfit and overfit (cases A and D). Whilst it may seem risky to ignore many of the p-values in each individual member of the holdout set, the GENDIFF measure does provide a robust measure of network generalization ability, if enough observations are used in the holdout set:

$$\text{GENDIFF} = \frac{1}{N_{cases}} \sum_{i=1}^{N_{cases}} \underset{j=1,\,\ldots,N_p \text{ s.t. } \{y_i\}_j \neq 0}{\text{median}} (\text{abs}(\{y_i\}_j - f_j(\{x_i\}))). \tag{3}$$

Figure 8 shows the GENDIFF value of the diagonal holdout set for the cantilevered beam case. Clearly, both the underfitting and overfitting regions of the network are penalized.

To summarize, the measure of generalization that will be used to decide when to terminate training is the median absolute error of the non-zero terms of the holdout set, averaged over every case in the holdout set. It is encouraging that the diagonal and line holdout sets show very similar behaviour, but it is necessary to decide which of the two holdout sets to use as the standard holdout set. Arbitrarily, the diagonal set is used, purely because the network has not seen any cases of the diagonal type during training.

3.5. EFFECTS OF NOISE ON NETWORK RESPONSE

Most real-world neural network applications perform network training using noisy data. The model updating application of neural networks has an advantage in that the training data set is guaranteed to be noise free, since it is obtained from the analytical model. However, the aim is to obtain a network that will respond sensibly to actual experimental data, which will be noisy. Consequently, it is essential that the response of the network to noisy input data is considered.

If there is a choice between two forms of a network that: (1) respond excellently to clean data but very poorly to contaminated data; or (2) respond reasonably well to noisy data but relatively badly to clean data, then clearly the network that responds best to noisy data is preferable. The previous section considered network training and testing using noise-free data. It will be shown in this section that the generalization performance of the network on clean data is not necessarily a good guide to its performance with noise contaminated data.

The GENDIFF termination criterion obtains the optimum training point with respect to a particular holdout set. As mentioned previously, the 200 observation diagonal holdout set is considered the standard holdout set. Figure 8 shows how, for the cantilevered beam the GENDIFF measure on the standard holdout set varied as each neuron was added. However, the standard holdout set is noise free and thus gives no indication of how well a given network responds to noise.

To consider the effects of noise on the generalization properties of the models, the input vectors of the holdout set were contaminated with various degrees of proportional Gaussian noise, as

$$\{x_i'\}_j = \{x_i\}_j(1 + r_{ij}), \quad j = 1, \ldots, N_p, \ i = 1, \ldots, N_{cases}, \tag{4}$$

where $\{x_i\}_j$ is the jth element of the ith input vector and r_{ij} is an observation from the normal distribution $N(0, \sigma^2)$.

Note that the corresponding target vectors remain clean, because the aim is to obtain the correct classification despite using noise contaminated data. The training data set also remains noise free. The degree of noise is set by the parameter σ, the standard deviation. A value of σ of 0·1 is defined as 10% noise. Figure 9 shows the GENDIFF measure for the T-shaped plate model, for eleven different Gaussian noise cases with the noise varying from 0 to 10% in 1% increments.

It can be seen that increasing the amount of Gaussian noise tends to increase the optimum GENDIFF value. This shows that the accuracy of the results obtained from the networks is reduced as the amount of noise increases, as expected. However, the degradation is gradual, seeming to vary linearly with the amount of noise. This is encouraging because if the addition of a small amount of noise caused the updating method to fail completely, then the method would be of no practical use. Many earlier updating methods suffered from this drawback [3].

A second point to note is that the optimum number of neurons decreased as the amount of noise applied to the holdout set increased. This is shown in Figure 10 (the solid lines) for both test models. The same qualitative behaviour was noted for both test cases, suggesting that the optimum number of neurons for noisy data is lower than that of noise-free data. However, it is dangerous to draw general conclusions when the only noise model that has been considered is proportional Gaussian noise. Proportional Gaussian noise has two properties that may aid the networks in dealing with the noise-contaminated data: firstly it is unbiased (zero mean), and secondly numerically small entries in the input vectors are not altered by significant absolute amounts.

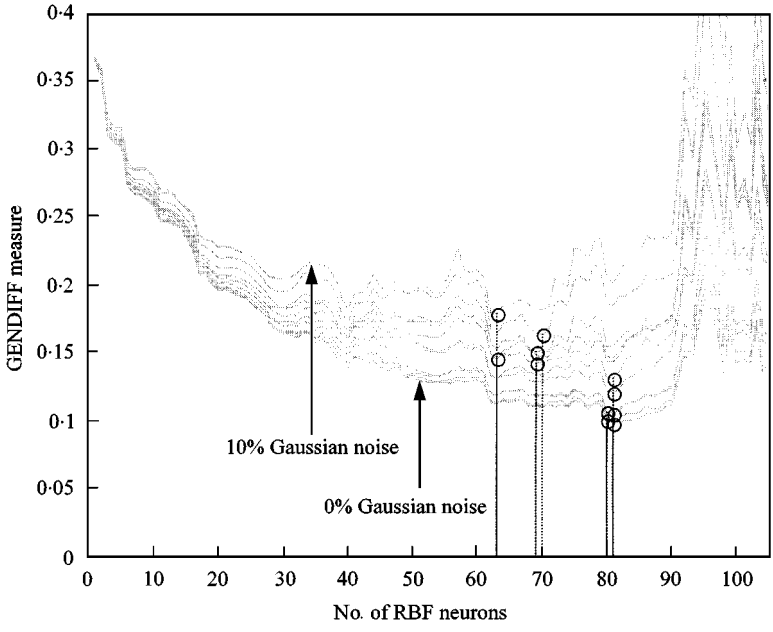

Figure 9. The effect of Gaussian noise on optimum generalization point, for the T-shaped plate: ——, generalization measure; ○, minimum location.

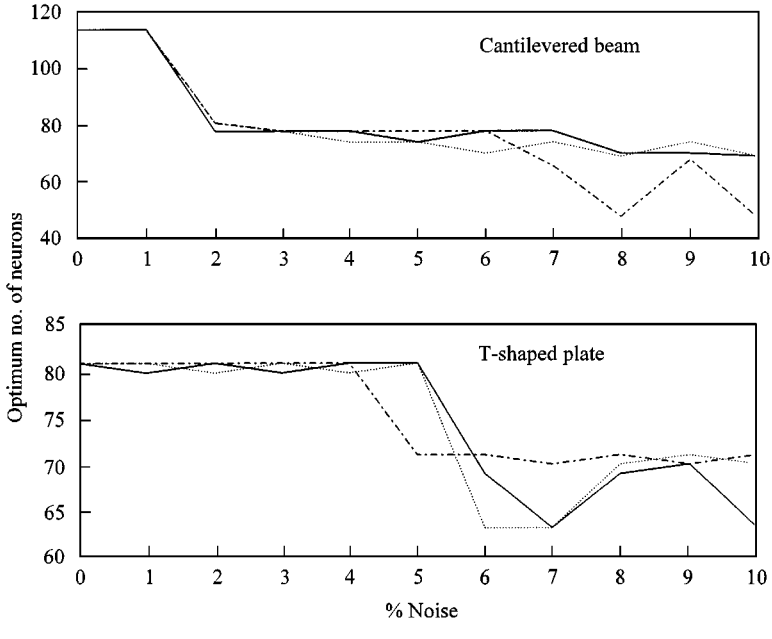

Figure 10. The effect of various noise types on optimum generalization point: ——, Gaussian noise; ·····, uniform noise; – – –, biased noise.

It is essential to consider noise models that do not have these helpful properties in order to assess the response of the neural networks to contaminated data more accurately. To consider biased noise, the same proportional Gaussian noise model is used but with the

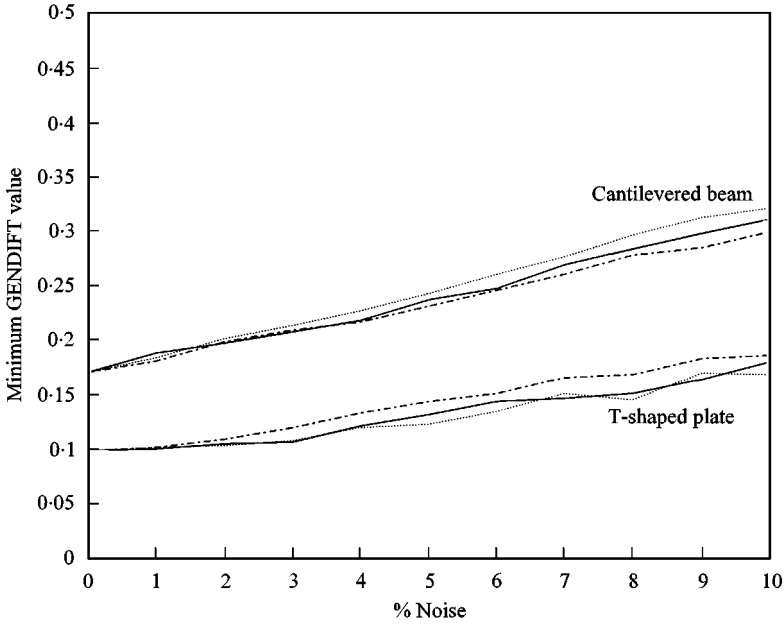

Figure 11. The effect of various noise types on optimum GENDIFF value: ——, Gaussian noise; ·····, uniform noise; – – –, biased noise.

absolute (i.e., positive) value of the term r_{ij} applied to the input vectors:

$$\{x_i''\}_j = \{x_i\}_j(1 + |r_{ij}|), \quad j = 1, \ldots, N_p, \ i = 1, \ldots, N_{cases}. \tag{5}$$

This form of noise has a mean of approximately $0{\cdot}8\sigma$; that is, as the percentage of noise increases, so does the bias of the noise. The final noise model considered is uniformly distributed noise, scaled by the largest value of the input vector. This form of noise applies a similar absolute error to all the terms. Note that the parameter σ (and hence percentage noise) in equation (6) does not represent the actual standard deviation for the noise case (analysis shows it is a factor of $\sqrt{12}$ too large):

$$\{x_i'''\}_j = \{x_i\}_j + u_{ij} \max(\{x_i\}), \quad j = 1, \ldots, N_p, \ i = 1, \ldots, N_{cases}, \tag{6}$$

where u_{ij} is an observation from the uniform distribution $U(-\sigma/2, \sigma/2)$.

The optimum number of neurons for the two extra noise models are also shown in Figure 10, as the dotted line and the dash–dotted line. Figure 11 shows the optimum GENDIFF value obtained for the three different noise models on the two test cases. It is striking how similarly the networks have responded to the different noise models, despite the difference between the response of the two test cases.

The three noise models are highly simplistic when compared with any noise that reality may produce. However, the fact that the networks have responded in a virtually identical fashion to the three different noise models suggests two useful properties:

(1) Neural networks are not sensitive to the *details* of the noise, but to the *distance* (in some sense) from the noisy input vector to the ideal noise-free input vector. This could be a highly useful property, because most noise models are not realistic; and

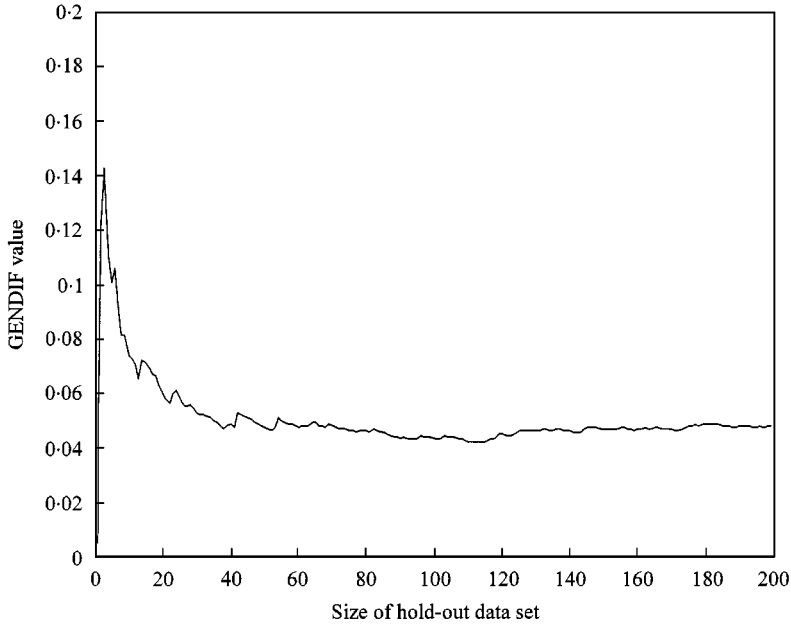

Figure 12. The variation of GENDIFF measure versus holdout set size.

(2) The optimum generalization point for a network can be obtained by considering the network response to data corrupted with proportional Gaussian noise.

Therefore, to recap:

- to obtain the optimum number of neurons, the standard holdout set is applied to the network with varying amounts of Gaussian proportional noise applied to the input vectors;
- the GENDIFF measure is calculated for each noise case as each new neuron is added to the net;
- the optimum number of neurons is obtained for each noise case; and
- the median of these values is used as the final termination point.

As an arbitrary compromise between accuracy and speed of calculation, the practical holdout test used to train networks will consider five different noise cases, with 2, 4, 6, 8 and 10% Gaussian noise.

The final point to consider about the use of holdout sets as a measure of generalization is the size of the holdout set. At the beginning of this section an arbitrary decision was made to use holdout sets containing 200 observations. It was simply hoped that this would be enough cases to model the generalization properties. To test whether this is true, Figure 12 shows how the GENDIFF metric varies for the noise-free case of the T-shaped plate at the optimum number of neurons, as the size of the holdout set is increased. It can be seen that the metric has nearly converged to a value after 60 observations have been considered, so a set of 200 seems more than sufficient. Similar behaviour was noted for the cantilevered beam holdout set.

3.6. WHOLE SET MEASURES OF GENERALIZATION

A disadvantage of the holdout method is that a significant amount of valuable training data is only used for network size selection, not for network training. There are two possible approaches commonly mentioned in the literature that allow the generalization ability of a network to be estimated whilst using all of the available data to train the network. These are information measures and cross-validation, and they will be considered below.

Information measures, such as the Akaike Information Criterion [4] and Bayes Information Criterion [5] typically use a formula to judge the "amount" of information that a network has seen in comparison to the number of weights (and hence generalization potential) that the network contains. However, these methods were derived for linear models and can perform inadequately for non-linear models. Initial tests performed on simple updating problems produced poor results, with the criteria not measuring the generalization ability of the networks. Considering the complexity of the model updating problem, it seems sensible to reject information criteria as a measure of generalization.

Cross-validation (CV) is another method for estimating the generalization error of a neural network without the use of a holdout set. To perform a cross-validation test the training data are randomly split into κ different subsets, where κ is decided in advance. The network is trained κ times, each time using $\kappa - 1$ of the sets as training data and the remaining set as a holdout set. The generalization error of each holdout set is calculated in turn, and combined to form an estimate of the generalization error of a network trained using all of the data.

If κ is equal to the number of training data observations, then the CV process is known as leave-one-out cross-validation. The CV process becomes more expensive to calculate as κ increases, although for single-output networks the leave-one-out cross-validation measure can be calculated in a more efficient manner using linear algebra theory [6].

Whilst the holdout approach gives an estimate of the optimum number of neurons, it is not clear how this information from the individual cross-validation networks can be combined. One possible approach is to calculate the optimum number of neurons for each CV network, using the GENDIFF measure on the particular holdout set (with Gaussian noise). If these κ results are averaged then a final network can be trained using all of the available data, with training terminating at this previously determined number of neurons. The final network will be trained using more training data than each of the individual CV networks. This may cause the choice of optimum number of neurons to be inaccurate (an underestimate), especially if κ is small.

Figure 13 shows the results obtained for the T-shaped plate when $\kappa = 20$. Each of the grey lines shows the GENDIFF results of one of the 20 CV runs. Clearly many of these runs show highly erratic behaviour, with poor generalization. These results can skew the mean GENDIFF value of the 20 runs, rendering it useless as a generalization estimate. The median is more robust to these outliers, and is also shown in Figure 13.

Cross-validation is rejected as a training method for the following reasons. Firstly, if a value of κ is used that is too low, then each CV network has significantly fewer training data observations than the total data set. This renders the estimate of the optimum number of neurons inaccurate. If κ is too high then the holdout sets are very small, causing many of the CV runs to be highly suspect.

Secondly, a great deal of computational effort is spent in training the multiple networks. For the two test cases considered, it is considerably cheaper to spend some of the computer time in generating a large holdout set. Additionally, if the computer time is spent in generating a large holdout set, then the neural network is tested over a greater sample of the underlying function. This increases confidence in the results.

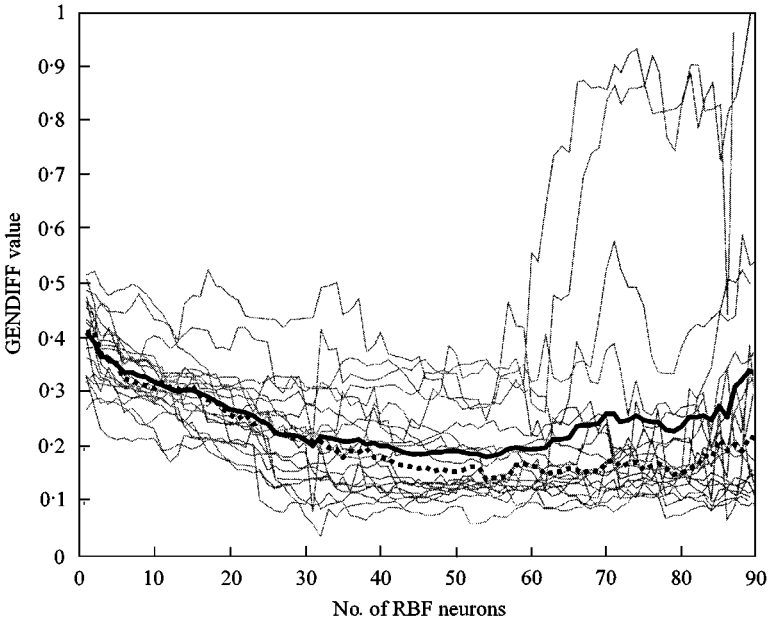

Figure 13. The results of $\kappa = 20$ cross-validation runs, for the T-shaped plate: ——, single cross-validation run; ——, mean of the 20 CV runs; ▪▪▪▪▪▪, median of the 20 CV runs.

Cross-validation is most appropriate when training data is hard to acquire, rendering holdout sets too expensive in terms of information withheld from the network. This is not the case for the model updating methodology considered here. It is possible that, for very large FE models, the CV approach will be cheaper than generating additional observations from the model. However, the approach will not be considered further for network training. Instead, the previous 200-strong diagonal holdout set will be used.

4. OPTIMUM NETWORK TRAINING STRATEGY

The remainder of this paper will be spent considering methods to improve the generalization ability of the neural networks. The previous sections established that a holdout set with proportional Gaussian noise using the GENDIFF metric is an appropriate way to obtain the optimum number of neurons for a given network. This section will consider possible improvements to the standard RBF network training method used previously.

Possible improvements to the RBF network include using alternative types of RBF neurons, and examining of the effects of the spread constant. Additionally, various alterations to the OLS training algorithm could be examined. Both of these approaches will be considered below.

Firstly, it is necessary to consider how to compare the generalization potential of different trained networks. The previous section established that holdout sets are a good method of measuring generalization potential. Unfortunately, the holdout set used to obtain the optimum training point of a network cannot then be used to estimate its generalization ability. This is because the network has in some sense "seen" the holdout data set, despite its absence from the training data. The holdout set would give an optimistically biased estimate of the generalization potential.

This necessitates the use of a third set of data to be used for network comparison purposes. The set used here will be another data set from the *diag* data set containing 1000 observations from the FE model. This should be an adequate number of cases to obtain a representative measure of the generalization ability of the network. The third data set, called the generalization set, will be presented to a trained network 5 times with varying degrees of Gaussian proportional noise applied to the input vectors. As before, the amounts of noise used will be 2, 4, 6, 8 and 10%. The GENDIFF values of the five different noise cases will be calculated. The mean of these five GENDIFF values will be termed the GEN value for the network, and will be used as an independent measure of the generalization ability of the network.

4.1. OPTIMUM SPREAD CONSTANT AND NEURON TYPE

Up until this point, all of the RBF neurons used in the first layer have used Gaussian radial basis functions, with the spread constant determined by the median Euclidean distance between the training data input vectors. This estimate of the spread constant should produce a reasonable network response, but it is possible that alternative values may be superior. Figures 14 and 15 show the GEN measurement resulting from training 100 different networks with different spread constant values, for the two test models. Each of the 100 networks was trained to the minimum GENDIFF criterion. The median, minimum and maximum Euclidean distances between any two of the training data cases are shown as vertical dotted lines.

These results show that the median estimate of the spread constant is not optimal for the cantilevered beam case. Indeed, optimal results for both cases are obtained by using higher than expected values of the spread constant. At these values, each Gaussian RBF neuron is active over a wide volume of the input space, so the neurons are not local. Since non-local Gaussian neurons have produced good results, this suggests that other non-local neuron types that do not require spread constants may be successful. The thin-plate spline function (equation (7)) is commonly used as an RBF function. The GEN value obtained by using the thin-plate spline function is shown in Figures 14 and 15 as the horizontal dotted line:

$$\phi(\upsilon) = \upsilon^2 \ln \upsilon. \tag{7}$$

The thin-plate spline function produced GEN results that were almost as good as the best result for the Gaussian neuron, without the need for a spread constant. This makes thin-plate spline RBF neurons superior to Gaussian neurons for model updating purposes. Note that there are many other non-local RBF functions that could be used, such as the cubic function (equation (8)) and even the identity function (equation (9)).

$$\phi(\upsilon) = \upsilon^3, \tag{8}$$

$$\phi(\upsilon) = \upsilon. \tag{9}$$

The identity function still produces a non-linear RBF neuron, because it is applied to the Euclidean distance between the input vector and the centre of the neuron. Table 2 shows the GEN results produced by all of the neuron types considered.

Whilst the thin-plate spline function produced adequate generalization in both cases, Gaussian neurons with the optimal spread constant produced better results. This optimal spread constant value could be found by training multiple networks using different spread constants, and selecting the optimum network afterwards. Unfortunately, this is a very

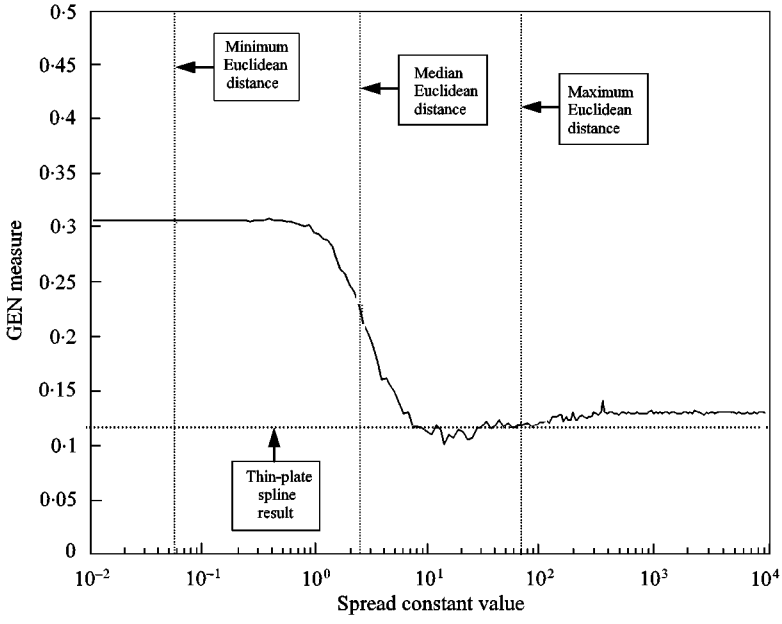

Figure 14. The effect of varying spread constant on generalization ability, for the cantilevered beam.

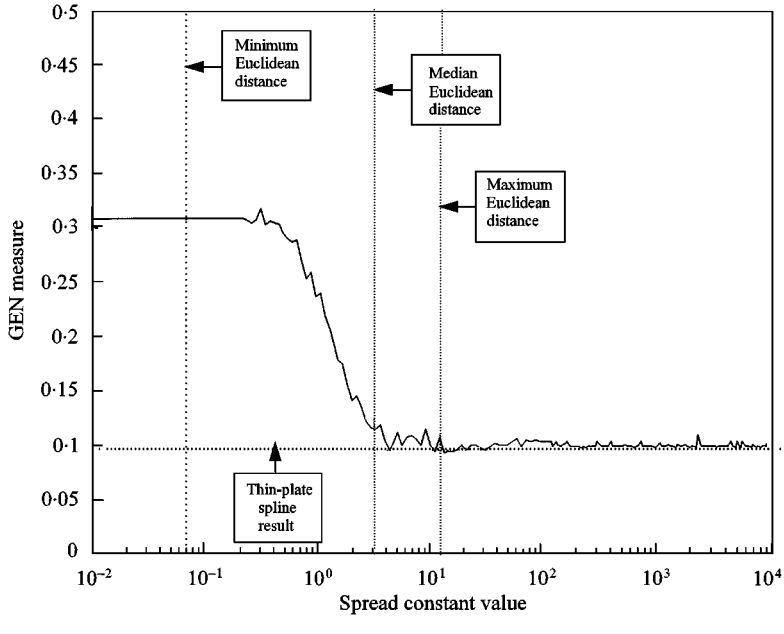

Figure 15. The effect of varying spread constant on generalization ability, T-shaped plate.

expensive technique, and most of the sub-optimal spread constant values are worse than the thin-plate spline neuron results. Also, the thin-plate spline function is more commonly used in the literature than the cubic or linear RBFs [2]. Consequently, the thin-plate spline function will be used as the standard RBF neuron.

TABLE 2

The generalization results obtained using different radial basis functions

Function type	GEN value, cantilevered beam	GEN value, T-shaped plate
Gaussian function Spread constant = initial guess	0·220	0·114
Gaussian function Spread constant = best value	0·102	0·0924
Thin-plate spline function	0·116	0·0978
Cubic function	0·130	0·100
Linear function	0·105	0·110

TABLE 3

The effect of including holdout data in second-layer training

Model	GEN value — initial	GEN value — holdout set included in second layer training
Cantilevered beam	0·1166	0·1093
T-shaped plate	0·0978	0·0907

The use of a holdout set with the thin-plate spline function means that the user does not have to set any parameters (e.g., spread constant, error goal) for network training; it is now a fully automated procedure.

4.2. IMPROVING RBF TRAINING ALGORITHM

This section will briefly consider an improvement to the OLS training algorithm. Firstly, it is noted that the holdout set is currently discarded after use. The members of the holdout set cannot be used as centres in the first layer of the RBF network — to do so would require an additional holdout set to judge the generalization ability of the new network, and this is not available. However, after the OLS algorithm has obtained the optimum first layer architecture, the holdout set *can* be merged with the training data for the linear optimization process that sets the weights in the second layer. Table 3 shows the difference resulting from this simple change.

This improvement requires virtually no extra time, and produces networks that are superior to the standard OLS algorithm. Accordingly, this technique will be considered part of the standard training algorithm for the remainder of this paper.

To summarize the results from this section, the standard network strategy consists of using an RBF network with thin-plate spline function RBF neurons in the first layer. This network is trained using the standard OLS algorithm, and the holdout set is merged with the training data set for the final second layer training step, after the network architecture has been fixed.

5. IMPROVING GENERALIZATION ABILITY

The neural network literature abounds with methods claiming to improve the generalization ability, robustness and training time of a given network [7]. It would be an unending task to test each of these methods. However, some of the approaches to improving generalization ability seem to have gained respectability and will be considered here.

There are two main approaches to preventing a given network from overfitting the training data, which is the main cause of poor generalization. These are structural stabilization and regularization. Structural stabilization involves selecting a network architecture that is not capable of overfitting the data, because it has the optimum number of parameters or "amount of freedom". This is actually the approach that has been considered earlier; the use of a holdout set to obtain the optimum network architecture is a form of structural stabilization.

The regularization approach involves selecting a model architecture that may be capable of overfitting the data, and constraining the weights of the network such that overfitting networks are discouraged. For example, many common regularization approaches penalize values of the weights that produce oscillating functions in favour of smooth functions. Another common method of performing regularization is to add a controlled amount of noise to the training data. This approach is called training with jitter. Whilst the addition of noise to clean training data may seem perverse, it has been shown [8] that the use of jitter can, in some cases, prevent the network from overtraining. This will be considered below.

5.1. TRAINING WITH JITTER

The use of jitter supposedly reduces overfitting by preventing the network training algorithm from getting a "lock" on the *exact* position of every point in the training data. It also seems reasonable that training the network with noisy data might improve the response to noisy inputs.

There are two types of jitter that may be applied to RBF network training. Both involve applying a degree of Gaussian proportional noise to the clean training data at various stages during training. The degree of noise applied is an important parameter because if too little jitter is applied then it will have little effect on training, but if too much is used then it will degrade the training data.

The first approach involves applying jitter to the clean training data before each additional neuron is selected, so the OLS algorithm sees a different set of training data at each step. The results of using various degrees of this form of jitter are shown in Figure 16, for both test cases. It is not clear that the use of jitter has improved the generalization ability.

The second form of jitter considered is only applied to the training data for training the second layer of the network. The first layer is always trained using noise-free data. If the underlying function is sufficiently smooth, then a small change to an input vector should not significantly perturb its corresponding target vector. This can be used to generate additional training cases from the original data set, by repeating the set several times and adding independent jitter to each of the copies. It is hoped that this enhanced set may result in improved generalization ability.

The results of applying this form of jitter to the cantilevered beam model are shown in Figure 17. This figure shows that increasing the amount of jitter simply degraded the network response. Additionally, making copies of the training data set appears to make little difference to the results. Given this poor performance of jitter-related training, this technique is not considered useful for this application.

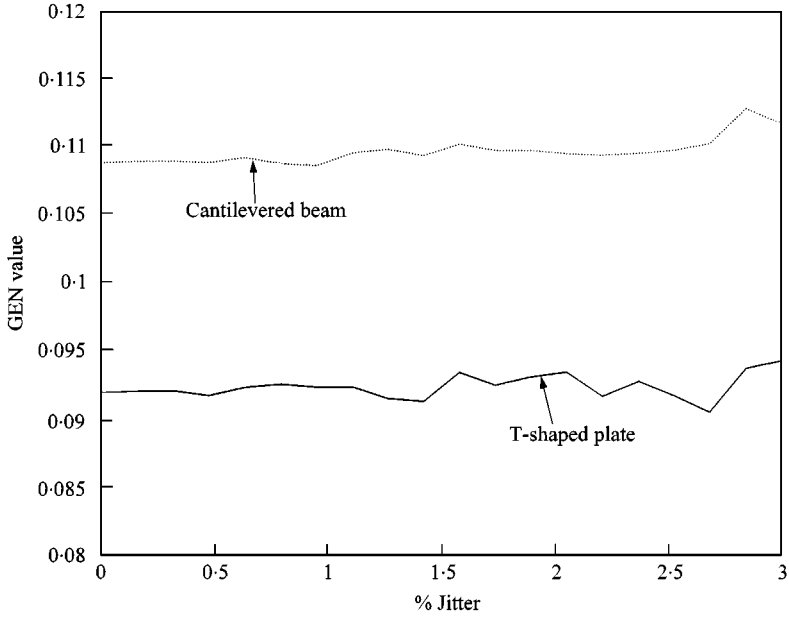

Figure 16. The results of applying jitter to training data for RBF layer training: ....., cantilevered beam; ——, T-shaped plate.

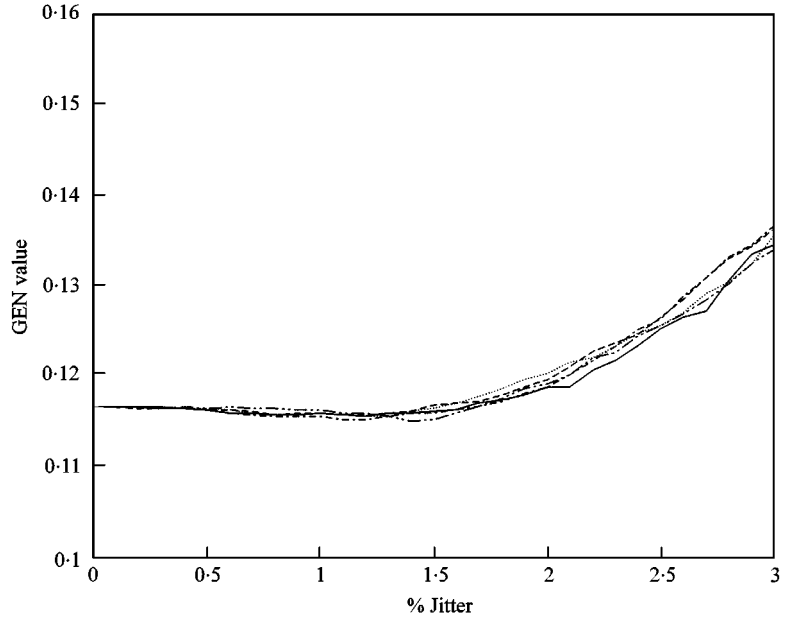

Figure 17. The results of applying jitter and repeating data sets for linear layer training, cantilevered beam: ——, 1 copy; ·....., 2 copies; - - - - -, 3 copies; - · - · - ·, 4 copies; - ·· - ··, 5 copies.

5.2. NETWORK COMMITTEES

The final method of improving the generalization potential of neural networks considered in this paper is the use of a committee of networks. A committee consists of multiple networks, each trained in a slightly different fashion. For example, some members of the committee might consist of networks trained using different training data, or termination criteria, or different radial basis functions or even different training algorithms. The output of the committee is composed of the mean of the output of each of the committee members. The response of a trained committee is shown in equation (10)

$$\eta(\{x\}) = \frac{1}{N_{comm}} \left\{ \sum_{i=1}^{N_{comm}} \eta_i(\{x\}) \right\}, \tag{10}$$

where N_{comm} is the number of members of the committee, η_i is the ith committee member (trained network) and $\eta(\{x\})$ is the committee response to the input vector $\{x\}$.

The generalization properties of various compositions of committees will now be tested. The first test considers generating committee members by altering the composition of the training and holdout sets. The standard RBF network, termination criteria and training algorithm are used. To generate the different committee members, the standard training and holdout data sets are pooled. Each new committee member is generated by assigning new holdout and training data sets randomly from the pool of data, with each set containing half of the available data.

Figure 18 shows the results for the cantilevered beam test case as more networks are added to the committee. The dashed line shows the generalization ability of each individual network. The solid line shows the GEN measure of the committee, as each new network is added. The results suggest that a committee of networks can significantly outperform

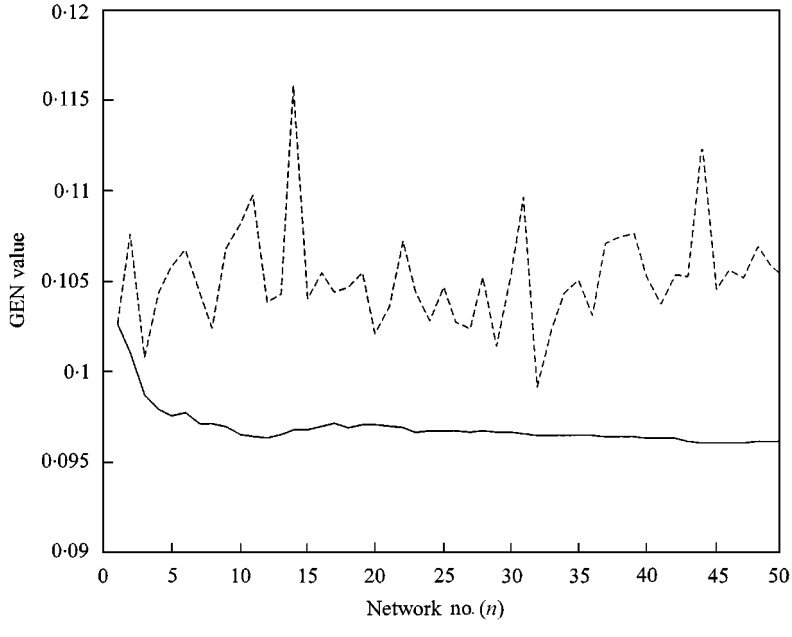

Figure 18. The use of committees, cantilevered beam: ----, nth network only; ——, mean response of first n networks.

TABLE 4

The generalization results obtained using committees

Model	Mean of the 50 Committee members individual GEN values	GEN value using mean of the output of all 50 members
Cantilevered beam	0·1051	0·0961
T-shaped plate	0·0890	0·0796

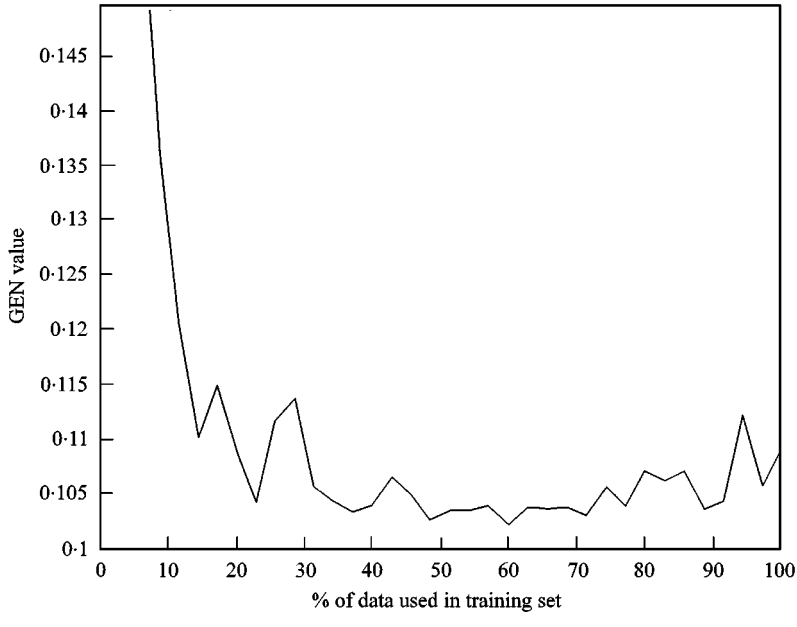

Figure 19. The effect of the composition of committees on generalization ability, cantilevered beam.

a single network. The solid line shows a clear, near-asymptotic downward trend as more networks are added. Similar results were obtained for the plate model. Table 4 summarizes the results, showing that the committees produced the best GEN values obtained in this paper. It seems from Figure 18 that the committee results stabilized as the size of the committee reached 10 networks. Hence, a committee consisting of around 20 networks should produce adequate results.

The final consideration is the ratio of the size of the training data set to the holdout data set. The 50/50 split used previously was arbitrary selected. If smaller training sets are used, then there will be significant variation between the committee members, and each member will have its optimum training point measured well. On the other hand, large training sets mean that each individual member is more accurate, but there is less variation between the members. It is not clear which approach will produce better results.

Figure 19 shows the generalization results of various compositions of the data sets for small committees (each committee shown contains five members). The data set composition does not seem to make a significant difference, as long as more than about 30% of the data is used in the training data set for each member. The optimum composition seems to use around 50–60% of the available data in the training set, so the initial estimate of 50% appears adequate.

6. CONCLUDING REMARKS

This paper has shown that measuring and improving the generalization ability of neural networks is not a trivial task. Throughout, the results and observations are based on examples. The standard measure of generalization ability, namely the mean-squared error of a holdout set, does not produce good results for model updating problems. A more robust measure was suggested, and shown to perform well in many tests throughout the paper.

The effect of noise on generalization ability was considered, and it was shown that neural networks perform well in the presence of noise. Additionally, the form of the noise did not seem important to the networks. This implies that the optimum training location may be obtained by considering a simple noise model such as Gaussian noise.

Various radial basis function types and training algorithms were considered. Significant improvements to generalization ability were noted by merging the holdout and training data sets for training the second layer of the network, after the network architecture has been decided. The Gaussian radial basis function was rejected as the radial basis function of choice, due to uncertainty regarding an appropriate value for the spread constant. It was noted that several alternative radial basis functions without spread constants gave excellent performance. The thin-plate spline neuron was used, primarily because of its common occurrence in the literature.

Finally, the use of jitter and committees to improve the generalization ability of networks was considered. It was found that jitter neither made an improvement nor degraded the results. It was also found that a committee of networks performed better than any single network. A good method of generating committee members is to split the available data evenly into multiple random holdout and training data sets.

ACKNOWLEDGMENTS

The authors would like to thank the Engineering and Physical Sciences Research Council for financial support and assistance.

REFERENCES

1. R. I. LEVIN and N. A. J. LIEVEN 1998 *Journal of Sound and Vibration* **210**, 593–607. Dynamic finite element model updating using neural networks.
2. C. M. BISHOP 1995 *Neural Networks for Pattern Recognition*. Oxford: Oxford University Press.
3. T. P. WATERS 1995 *Finite element model updating using frequency response functions. Ph.D. Thesis, Department of Aerospace Engineering, University of Bristol, U.K.*
4. W. S. SARLE 1995 *Proceedings of the 27th Symposium on the Interface of Computing Science and Statistics*, 352–360. Stopped training and other remedies for overfitting.
5. A. E. RAFTERY 1995 *Bayesian model selection in social research. Sociological Methodology* (P. V. MARSDEN, editor). Cambridge, MA: Blackwell.
6. M. J. L. ORR 1996 *Introduction to Radial Basis Function Networks*. URL: http://www.cns.ed.ac.uk/people/mark/intro.html, University of Edinburgh.
7. W. S. SARLE (editor) 1998 *Neural Network FAQ*. Periodic posting to the Usenet newsgroup *comp.ai.neural-nets*. URL: ftp://ftp.sas.com/pub/neural/FAQ.html.
8. G. AN 1996 *Neural Computation* **8**, 643–674. The effects of adding noise during backpropagation training on generalization performance.