Balancing Bias and Variance: Network Topology and Pattern Set Reduction Techniques

T.D. Gedeon [1,3], **P.M. Wong** [2] and **D. Harris** [4]

[1] School of Computer Science and Engineering
[2] Centre for Petroleum Engineering
The University of New South Wales
Sydney, 2052. AUSTRALIA

[3] Department of Telecommunications and Telematics
The Technical University of Budapest
Sztocek u. 2, H-1111, HUNGARY

[4] Centre for Neural Networks
Kings College, London, UK

Abstract

It has been estimated that some 70% of applications of neural networks use some variant of the multi-layer feed-forward network trained using back-propagation. These neural networks are non-parametric estimators, and their limitations can be explained by a well understood problem in non-parametric statistics, being the "bias and variance" dilemma. The dilemma is that to obtain a good approximation of an input-output relationship using some form of *estimator*, constraints must be placed on the structure of the estimator and hence introduce bias, or a very large number of examples of the relationship must be used to construct the estimator. Thus, we have a trade off between generalisation ability and training time.

We overview this area and introduce our own methods for reducing the size of trained networks without compromising their trained generalisation abilities, and to reduce the size of the training pattern set to improve the training time again without reducing generalisation.

Model Assumptions

In this paper, we will assume a multi-layer feed-forward network trained using back-propagation [1], and will use the general expression "neural network" to mean such a network. All connections are from units in one level to units in the next level, with no lateral, backward or multi-layer connections. Each unit is connected to each unit in the preceding layer by a simple weighted link. The network is trained using a training set of input patterns with desired outputs, using the back-propagation of error measures. The network is tested using a validation set of patterns which are never seen by the network during training and thus can provide a good measure of the generalisation capabilities of the network. The separation of the total set of patterns into training and test sets is generally at random to avoid introducing experimenter bias.

By back-propagation we mean the general concept of developing the error gradient with respect to the weights, and not restricted to the original gradient descent method. In the examples we use here, we have used the basic sigmoid logistic activation function, $y = (1 + e^{-x})^{-1}$, though this is not essential to the substance of our results.

Statistical Background

Non-parametric statistics is concerned with model free estimation. When employed for classification both parametric and non-parametric statistics seek to construct decision boundaries between the various

classes using a collection of training samples. Non-parametric methods differ from parametric methods in that there is no particular structure assigned to the decision boundaries, *a priori*.

The obvious advantage of parametric techniques is efficiency. By setting the structure of the decision boundary before estimation begins then fewer data points (or training examples) are required. This is because there are (hopefully) a small number of parameters in the parametric model that require estimation. Non-parametric or model free estimation potentially requires the estimation of an infinite number of parameters and hence needs a much larger number of training examples. However, the efficiency of parametric methods comes at a cost. If the actual form of the decision boundary departs substantially from the assumed form, then parametric methods can result only, in the 'best' approximation for the decision boundary from within the adopted class of decision boundaries. Non-parametric methods place no restriction on the class in which the decision boundary used in estimation must reside.

Informally, *consistency* is the asymptotic convergence of the estimator to the object of estimation. In this context asymptotic refers to the sample size or the number of patterns in the training set approaching infinity. Most non-parametric algorithms are consistent for any regression function $E[\mathbf{y}|\mathbf{x}]$[2]. Indeed it has been shown [2, 3] that feed forward networks are consistent, under appropriate conditions relating to the architecture of the network. Consistent in the sense that the weights in the network will, in the limit of training set size approaching infinity, converge to the optimal weights $\mathbf{w}^*$. Although this is an encouraging property, non-parametric methods can be very slow to converge, and this has indeed been observed in the training times for neural networks.

Non-parametric estimators are guaranteed to perform optimally in the limit. In the context of neural networks, they are only guaranteed to outperform other parametric estimators when the size of the training set approaches infinity. For a finite sample, non-parametric estimators can be very sensitive to the actual realisations of $(\mathbf{x},\mathbf{y})$ contained in the sample. This sensitivity results in an estimator that is high in what is known as *variance*. The only way to control this variance is to introduce some *a priori* structure into the estimator, that is to use parametric methods. This approach also has its pitfalls. In complex classification problems, it is difficult to know the structure to impose on the estimator. As mentioned above, this can result in estimators that converge to an incorrect solution. This creates models that are high in what is known as *bias*. The performance function used in back-propagation can be readily decomposed into a bias and a variance term [4, 5].

It is this dilemma between bias and variance that can explain the limitations of non-parametric learning. Low bias and low variance requires large numbers of training examples. In situations where it is not possible to obtain sufficiently large numbers of training examples it is necessary to allow some bias into the training procedure. In the next section we provide a brief overview of the decomposition of neural network training into bias and variance terms.

Bias-variance decomposition

The neural network is solving a regression problem, so we construct a construct function based on the training set:

$$f(\mathbf{x}; D) \qquad f \text{ depends on training data } D$$

We use the mean squared error of f as an estimator of regression:

$$E_D\left[(f(\mathbf{x}; D) - E[y \mid \mathbf{x}])^2\right]$$

The bias-variance decomposition of the estimator is:

$$\begin{aligned} &(E_D[f(\mathbf{x}; D)] - E[y \mid \mathbf{x}])^2 && \text{"bias"} \\ &+ E_D\left[(f(\mathbf{x}; D) - E_D[f(\mathbf{x}; D)])^2\right] && \text{"variance"} \end{aligned}$$

The interpretation of the above is as follows:

- If the expected value of the function is different on average from the regression, then it is biased.
- If the function has large differences from the expected value it has high variance.

We do not in general have access to the estimator function learned by the neural network, the derivation of the function encoded by neural network weights is an entire area of research. Nevertheless, the estimator function can be approximated, over a number of test patterns over a number of trained neural networks. We chose the test patterns to be 1/3 at random of the available patterns.

The bias variance decomposition estimated bias (over the set of test patterns) is:

$$\text{Bias}\{x\} \approx \frac{1}{test} \sum_{p=1}^{test} \bar{f}(x_p, w) - y_i \quad ^2$$

The estimated variance is calculated over the test patterns over 50 different trained neural networks, with each network being trained on a randomly selected 3/4 of the patterns from the remaining 2/3 of the original pattern set available. Each network is thus trained using 1/2 of the original patterns. The estimated variance is:

$$\text{Variance}\{x\} \approx \frac{1}{test} \sum_{p=1}^{test} \frac{1}{50} \sum_{p=1}^{50} f(x_p, w; D_i) - y_i \quad ^2$$

Network topology reduction – pruning

The introduction of bias into a trained neural network by reducing the number of processing units is generally referred to as pruning. Although the output units perform computations, the elimination of one or more of these units modifies the nature of the output function being learnt by the network, hence we will restrict ourselves to considering the case of hidden units only.

The seminal work on pruning trained networks [6] uses the outputs of units in a two stage pruning process, which operates by inspection. Such pruning by inspection is difficult even on small examples, some automatable process would be ideal. Properties such as *relevance* [7, 8], *contribution* [9], *sensitivity* [10], *badness* [11], and *distinctiveness* [12] have been described in detail elsewhere. We will briefly describe *distinctiveness* here.

The *distinctiveness* of hidden units is determined from the unit output activation vector over the pattern presentation set [12]. That is, for each hidden unit we construct a vector of the same dimensionality as the number of patterns in the training set, each component of the vector corresponding to the output activation of the unit. This vector represents the functionality of the hidden unit in (input) pattern space. In this model, vectors for clone units would be identical irrespective of the relative magnitudes of outputs and recognised. Units with short activation vectors in pattern space are recognised as insignificant and removed.

Pattern	1	2	3	4	5	6
p.000	1.000	1.000	1.000	1.000	1.000	0.000
p.001	0.000	1.000	1.000	0.000	0.000	0.000
p.002	0.000	0.000	0.000	1.000	0.000	1.000
p.003	0.000	0.000	0.000	1.000	1.000	0.000
p.004	0.000	0.000	1.000	1.000	0.000	0.000
p.005	1.000	0.439	1.000	1.000	0.999	0.706
p.006	0.000	1.000	0.000	1.000	1.000	0.000
p.007	1.000	0.000	0.000	1.000	0.000	0.000
p.008	1.000	0.000	0.000	1.000	0.000	0.000
p.009	0.000	0.000	0.000	0.000	0.000	0.000
p.010	0.000	0.000	1.000	0.000	0.167	0.000
p.011	0.000	0.000	1.000	0.000	0.000	0.000
p.012	0.000	0.000	0.000	0.000	0.000	0.000
p.013	1.000	0.000	0.000	0.989	1.000	1.000
p.014	0.000	0.000	0.000	0.000	0.000	0.000
p.015	0.000	0.000	1.000	1.000	1.000	0.000

Table 1. Six hidden unit activations by pattern.

The recognition of similar pairs of vectors is done by calculation of the angle between them in pattern space. Since all activations are constrained to the range 0 to 1, the angle calculations are normalised to 0.5, 0.5. Angular separations of up to about 15° are considered too similar and one of them is removed. The weight vector of the unit which is removed is added to the weight vector of the unit which remains. With low angular separations, the averaging effect is minor and the mapping from weights to pattern space remains adequate in that the error measure is no worse subsequently. This produces a network with one fewer unit which requires no further training. Similarly, units which have an angular separation over about 165° are complementary, and both can be removed. It would be possible to discover pairs of similar units by inspection of the above table, but would be difficult for much larger numbers of patterns or units. The vector angles for the six hidden units range from 61.6° to 90°. Clearly, none of the units are similar to any other.

A further category of undesirable units is also discovered and included in the distinctiveness analysis, which is not found by the other methods. Groups of three or more units which together have no effect, or two or more units with a constant effect can be recognised. That is, in the pattern space, the sum of their vectors is zero or constant. The discovery of such groups is done by a sorted Gaussian vector pivot on the cumulative

rectangular matrix of pattern space vectors. This produces the reduced row echelon matrix. For the case of groups of units with jointly no effect, the entire group can be removed. If the joint effect is constant, a bias can replace the entire group. Because of the inaccuracies incurred by banding and subsequent use of this information, it may be necessary to retrain the network. Fortunately, such groups are not common in our experience, therefore retraining is not often required.

Reducing the size of training sets

During training, frequency distributions of the errors for all patterns in the training set show that very early in training, the distribution of the pattern errors is approximately normal with a large variance. The network then dramatically reduces the errors for the majority of the training set. There remain patterns with relatively high error, creating an almost bimodal error distribution, with the low error peak containing patterns the network has learnt well, and the high error peak containing the outliers. From the two peaks in the error distribution it is clear that the network can identify outliers itself. It is difficult and time consuming to identify a bimodal error distribution during training. Fortunately, a measure of the variance achieves the same effect.

Patterns should not be removed too quickly, as those patterns with midrange errors could eventually be learnt by the network. Bimodal Distribution Removal (BDR) is intentionally conservative in its removal of patterns to give the network opportunity to learn these *slow coaches* [5].

Should BDR be continued indefinitely, eventually all the patterns would be removed from the training set. This is undesirable because as the training set becomes smaller the network is devoting a large number epochs of training to a reduced set of examples, thus potentially dramatically increasing the overfitting effect. Removal of the outliers from the training set causes the high error peak to shrink – this could be used as a halting condition for training.

BDR attempts to address the usual weaknesses of outlier detection and removal methods in that:

- pattern removal does not start until the network itself has identified the outliers,
- the number of patterns removed is not hard wired, but instead is data driven, and
- patterns are removed slowly, to give the network time to extract information from them.

This method was tested by calculating the estimated bias and variance terms over the test set, over a number of networks being trained. Two other methods were tested as controls. The first of these is the Least Trimmed Squares (LTS), which minimises only the lowest mean square errors, hence the outliers never used which leads to better generalisation, however we must know how many outliers exist [13]. The second method is the Heuristic Pattern Reduction (HPR) method, which is not an outlier reduction method *per se*, but attempt to reduce the overall size of the training set by mapping the individual training patterns onto a one-dimensional adjacency measure using their contribution to the total sum of squares. This adjacency is used as the base to make arbitrary assumptions as to sizes of clusters of training patterns. Thus, assuming an homogenous distribution of clusters of pairs of patterns, the training set can be reduced to half its size [14].

One third of the patterns were set aside as a test set and were never used in training. The remaining patterns were used to create 50 sets of pattern training sets at random. Each of these used about 3/4 of the remaining 2/3 of the original patterns. Fifty networks for each of methods BDR, LTS and HPR as described above were trained, as well as normal back-propagation. The estimated bias and variance were then calculated.

Least Trimmed Squares provided some control of variance at the expense of higher bias. This control is significant if training is continued for a long time (since the overfitting effect increases). The LTS method to control variance requires *a priori* knowledge of the amount of noise in the training set. Heuristic Pattern Removal produces an surprising result. We have discussed that neural performance becomes optimal as the size of the training set approaches infinity. Yet, measurements of bias and variance for training on a half size training set showed that the Heuristic method performs almost as well as the Bimodal Distribution Removal method. Bias and variance are very sensitive to the complexity of the data and by how much the training set is reduced. Eventually, the Heuristic method lead to the most uncontrolled increase in variance of all the methods. Bimodal Distribution Removal provides a similar control of variance to LTS. It is an improvement over LTS since both bias and variance are lower during training, and we do not need the *a priori* knowledge of the number of outliers. The problem remains in finding the 'correct' time to halt removal of further points.

The values for bias in the LTS and BDR methods for this data set in comparison to normal back-propagation indicates that even though these methods perform well, the noisy data points are being useful in

this case. This means our implicit assumption that the probability law γ is approximately degenerate is barely valid. This points to the requirement for appropriate choice of a data set.

Both the HPR and BDR methods require some means of determining when to terminate the training set reduction. The benefits of these methods derive from the use of the neural network behaviour to help determine which points are outliers. Both of these methods use static measures, hence we developed a method to utilise the dynamics of the network behaviour during training on training patterns, called error sign testing.

Error Sign Testing

During the training phase each input vector of the training set is presented to the network and an output vector is obtained. An error vector is then calculated by taking the difference between the desired and the output vectors. The magnitude of the error vector can be used as a measure of how easily the input pattern is learnt in a given epoch, the lower the value the easier to learn. For successful learning of a good pattern, the error magnitude for a particular pattern in the n^{th} epoch of batch training, say E_n, should be smaller than the previous one, E_{n-1}. Therefore, counting the number of negative signs of the expression E_n-E_{n-1} for K epochs can be used to define a *good* pattern. In order words, a pattern with a large percentage of negative signs will be a good pattern. Similarly, a small percentage of negative error signs (or large percentage of positive error signs) will be a noisy pattern or outlier. This method of detecting outliers is called Error Sign Testing (EST).

The value of K can be determined by the root-mean-square-error (rmse) of all the training patterns presented to the network. In most applications, the rmse starts from a high value and drops very quickly in a small number of epochs. In this example study, K was determined at an epoch when oscillation of rmse began, say 200. This is due to the presence of the outliers, and hence the learning of good patterns slows down and the network starts to overfit the outliers.

After K epochs, the percentage of negative signs of each pattern is calculated. In this study, a pattern which has less than 10% of the negative error signs is defined as an outlier. The outliers are then removed and further training is performed using a reduced pattern size. A validation data set is also used to test the performance of the trained network, but is not used for training, nor are any outliers removed from this test set. An example study using data from an oil well in a real reservoir will be used to demonstrate our method.

Example Study

Our technique for pattern reduction was used on a data set from the wireline logs and core data in an oil well located at the North West Shelf, Australia. The data set consists of points along the well with 4 wireline log measurements. The log variables in this well included deep resistivity (RLLD), bulk density (RHOB), sonic travel time (DT) and gamma ray (GR). The rock type of each of the points was identified through the careful examination by a geologist of the core sample taken at that location. In this study, five dominant rock types were found and were then named as rock types 1 to 5 for simplicity purposes. A cross-plot of RHOB versus DT is shown in Figure 1. The core sampling process is not usually done on every well and the predictions of rock types must then rely on the available log data. Neural network methods have found some recent applications in well log analysis [15, 16].

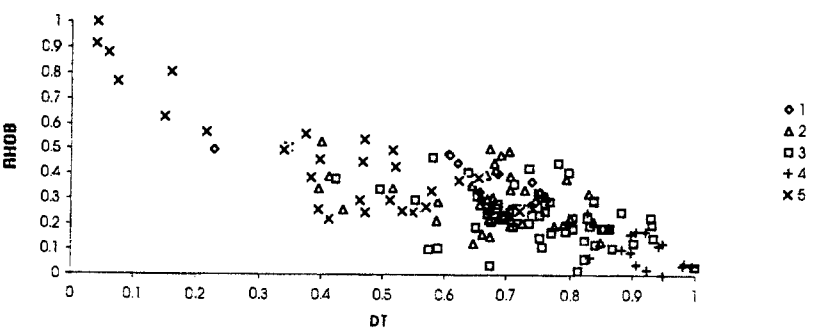

Figure 1. Relationships of RHOB and DT for different rock types

In this study, the four log measurements and the corresponding rock types were used as the input patterns and the desired output patterns respectively. The network architecture used has four input units corresponding to each of the log variables, and four units in the hidden layer. Each rock type was represented by one output unit, and hence five output units were used. The classification was considered to be correct if the outputs of the network were within 0.1 of the desired value of 0 or 1. The network configuration used produced acceptable predictions of the rock type. By acceptable, we meant the results from neural network was a consistent improvement compared to those obtained from the standard statistical techniques.

Experiments and Results

The log measurements usually contain noisy data, especially in a lithologically complex reservoir. However, the outliers are not easily recognised in most of the cases (refer to Figure 1). Note that this does not include potentially unbounded outliers, which are clearly markedly different to the normal data and are excluded using standard methods. This exclusion of markedly different outliers is necessary for our method, as it is specialised to the location of any remaining outliers, and the sum of squares error term is not sufficiently sensitive to compensate for very extreme values which could pull the network towards them and still reduce the overall squared error. In this study, the proposed error sign testing method was used to detect and remove any remaining outliers. Four experiments with different training sets were trained and the performance of each set (i.e. classification accuracy) was evaluated using the validation test data. The structure of each training set is described as follow:

(1) whole data set - this data set consisted of the original data which was used as a control experiment.
(2) half of the whole data set - this data set was selected based on the percentage of negative error signs during training using each of the original patterns. The patterns were then sorted in ascending order and the new training set was formed using every second sorted pattern. The aim of this training set was to remove half of the good and noisy data, in analogy with the HPR method described earlier.
(3) clean data set - the outliers, defined by the EST method, were removed from the original data set. After doing this, the data set was then considered to be 'clean'.
(4) half of clean data set using error sign testing method - this data set was formed using the clean data set in (3) followed by the half reduction method using the EST method as in (2). This experiment was designed to further simplify the set of patterns in (3).

Each of the above experiments was performed with 10 different sets of initial weights, and was trained for 10,000 epochs. In each run, the same initial weights are used for training the original data set (1), and the corresponding sets of reduced patterns (2), (3) and (4). This was done in order to minimise the effects of the initial random functionality of the network unit weights. The validation set was also used to record the highest classification accuracy (in percentage) during the training phase.

The results of the comparison study are tabulated in Table 1. Note that the classification accuracy shown are the maximum values of the number of runs done and can be from different runs. The average training times (measured in number of epochs) required to achieve the maximum accuracy in each experiment are also shown. The success of the method is shown by the high classification accuracy on the validation set. Statistical evaluation of the same data using discriminant analysis was also done on the whole training set. The classification accuracy was only 57% on the validation data. Therefore, the results using neural network methods showed better performance in generalisation.

Training Set	Average Training Time (epochs)	Classification Accuracy (%)
(1) Whole Data Set	3500	66
(2) Half of Whole Data Set	3400	64
(3) Clean Data Set	2900	66
(4) Half of Clean Data Set	1400	68

Table 1: Performance of Different Training Sets on the Validation Data.

The results on this example also showed that a smaller set of training patterns did not necessarily degrade the generalisation ability of the trained network. Pattern reduction techniques aim to simplify the error surface of the pattern space, however simple elimination of half of original training patterns did not show significant improvement. This is most likely due to the presence of half of the original outliers in the remaining training data set, and these outliers may affect the process of error surface simplification. When the outliers were defined and removed from the training set, the same classification accuracy was obtained compared to the whole data set.

The clean data set was also reduced to half its size and the results showed better performance in generalisation. This improvement was probably due to the simplification of the error surface in the pattern space. Figure 2 shows the outliers identified by the EST method in the run with maximum classification accuracy. In this case, 12 outliers were found in rock types '2' and '3', and they were coded as '2x' and '3x'. As displayed in Figure 2, the outliers defined tend to lie within the clusters of other rock types, and therefore including these data in the training set will significantly increase the training time, and overfitting of these outliers will also occur.

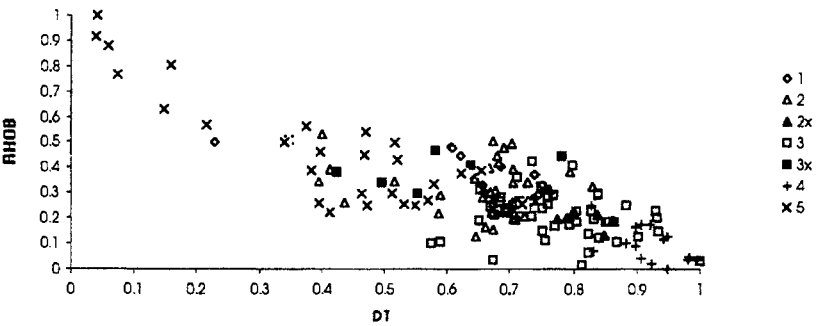

Figure 2. Outliers (coded as '2x' and '3x') identified by the Error Sign Testing Method

The removal of outliers did not seem to improve generalisation, however the clean data set took less training time to achieve the same results as obtained from the original data set. Therefore removing outliers in the training set does reduce the amount of training time. Note that the above time are shown in epochs, yet the set (4) is less than half the size of set (1), and hence the training time of set (4) versus set (1) is reduced by a factor of over 5. This result can be explained in terms of the bias-variance balance. The removal of half of the patterns increases the bias, as the network will now be more sensitive to the particular patterns that happen to be represented in the smaller training set, rather that to the underlying function we are attempting to approximate. This is balanced by the reduction in variance we obtain from correctly removing the outliers in the training set. Since the results are statistically the same between sets (1) and (4), we can therefore be convinced that the EST method has indeed removed the correct outliers.

Finally, it is worth mentioning that the neural network method in this study performed better than the standard statistical approach. However, the classification accuracy obtained from discriminant analysis can be used to provide a minimum baseline level of accuracy for the neural network method to achieve.

Conclusions

We have discussed neural networks in term of non-parametric estimators, and how their limitations can be explained by a well understood problem in non-parametric statistics, being the "bias and variance" dilemma, which for neural networks is essentially a trade off between generalisation ability and training time.

We have described some approaches to control the bias and variance balance in a neural network during training, methods for reducing the size of trained networks without compromising their trained generalisation abilities, and to reduce the size of the training pattern set to improve the training time again without reducing generalisation. We have also discussed how to measure the bias and variance in a number of neural networks during training. We have introduced a method of outlier detection, and indicated how the balance of bias and variance can be used to come to some qualitative understanding of the usefulness and correctness of the outlier detection algorithm.

References

1. Rumelhart, DE, Hinton, GE, Williams, RJ. "Learning internal representations by error propagation," in Rumelhart, DE, McClelland, *Parallel distributed processing*, Vol. 1, MIT Press, 1986.
2. White, H, "Learning in artificial neural networks: A statistical perspective," *Neural Computation*, vol 1., pp. 425-464, 1989.
3. Gallant, AR and White, H, *A Unified Theory of Estimation and Inference for Nonlinear Dynamic Models*, Basil Blackwell, Oxford, 1988.
4. Geman, S, Bienenstock, E and Doursat, R, "Neural networks and the bias/variance dilemma," *Neural Computation*, vol. 4, pp. 1-58, 1992.
5. Slade, P and Gedeon, TD "Bimodal Distribution Removal," *Proceedings IWANN International Conference on Neural Networks*, Barcelona, 1993. also is Mira, J, Cabestany, J and Prieto, A, *New Trends in Neural Computation*, pp. 249-254, Springer Verlag, *Lecture Notes in Computer Science*, vol. 686, 1993.
6. Sietsma, J, & Dow, RF, "Neural net pruning - why and how," *Proc. Int. Joint Conf. on Neural Networks*, vol. 1, pp. 325-333, 1988.
7. Mozer, MC, Smolenski, P, "Using relevance to reduce network size automatically,", Connection Science, vol. 1, pp. 3-16, 1989.
8. Segee, BE, & Carter, MJ, "Fault Tolerance of Pruned Multilayer Networks," *Proc. Int. Joint Conf. on Neural Networks*, vol. 2, pp. 447-452, Seattle, 1991.
9. Sanger, D, "Contribution analysis: a technique for assigning responsibilities to hidden units in connectionist networks," *Connection Science*, vol. 1, pp. 115-138, 1989.
10. Karnin, ED, "A simple procedure for pruning back-propagation trained neural networks," *IEEE Transactions on Neural Networks*, vol. 1, pp. 239-242, 1990.
11. Hagiwara, M, "Novel back propagation algorithm for reduction of hidden units and acceleration of convergence using artificial selection," *IJCNN*, vol. 1, pp. 625-630, 1990.
12. Gedeon, TD, Harris, D, "Network Reduction Techniques," *Proc. Int. Conf. on Neural Networks Methodologies and Applications*, AMSE, San Diego, vol. 2, pp. 25-34, 1991.
13. Joines, M and White, M, "Improving generalisation by using robust cost functions," *IJCNN*, vol. 3, pp. 911-918, Baltimore, 1992.
14. Gedeon, TD and Bowden, TG, "Heuristic Pattern Reduction," *Proc. Int. Joint Conf. on Neural Networks*, vol. 2, pp. 449-453, Beijing, 1992.
15. Rogers, SJ, Fang, JH, Karr, CL and Stanley, DA "Determination of Lithology from Well Logs using a Neural Network," *AAPG Bulletin*, 76, p. 731-739, 1992.
16. Wong, PM, Gedeon, TD and Taggart, IJ "An Improved Technique in Porosity Prediction: A Neural Network Approach," *IEEE Trans. Geoscience and Remote Sensing*, (in press), 1994.