ANALISIS 6 (1&2), 75-87 (1999)

IMPROVING GENERALIZATION OF NEURAL NETWORKS USING LENGTH AS DISCRIMINANT

FADZILAH SIRAJ & DEREK PARTRIDGE

ABSTRACT

This paper discusses the empirical evaluation of improving generalization performance of neural networks by systematic treatment of training and test failures. As a result of systematic treatment of failures, a discrimination technique using LENGTH was developed. The experiments presented in this paper illustrate the application of discrimination technique using LENGTH to neural networks trained to solve supervised learning tasks such as the Launch Interceptor Condition 1 problem. The discriminant LENGTH is used to discriminate between the predicted "hard-to-learn" and predicted "easy-to-learn" patterns before these patterns are fed into the networks. The experimental results reveal that the utilization of LENGTH as discriminant has improved the average generalization of the networks increased.

ABSTRAK

Kertas ini membincangkan penilaian empirik dalam meningkatkan pencapaian rangkaian buatan dengan memberi rawatan kepada corak yang gagal selepas pembelajaran dan pengujianan. Hasilnya, teknik pembeza layan dengan menggunakan LENGTH telah dihasilkan. Eksperimen yang dibincangkan dalam kertas kerja ini mengilustrasikan aplikasi teknik pembeza layan menggunakan LENGTH bagi melatih rangkaian neural dalam menyelesaikan tugas pembelajaran berbantu seperti masalah "Launch Interceptor Condition 1". Pembeza layan LENGTH digunakan untuk membezakan antara corak ramalan "hard-to-learn" dan "easy-to-learn" sebelum corak tersebut dilatih oleh rangkaian. Dapatan kajian menunjukkan bahawa penggunaan pembeza layan LENGTH telah meningkatkan purata pencapaian rangkaian neural.

INTRODUCTION

The goal of network training is not to learn an exact representation of the training data itself, but rather to build a statistical model of the process which generated the data. This is important if the network is to exhibit good generalization, that is, to make a good prediction for the new inputs. Multilayer perceptrons (MLP) are now widely used for problem recognition, although the training remains a time consuming procedure and often converging toward a local minimum. Moreover, as the optimum network size and topology are usually unknown, the search of this optimum requires a lot of networks to be trained.

Webb and Lowe (1990) illustrated that a nonlinear adaptive feed-forward layered network with linear output units can perform well as a pattern classification device. They have also shown that the discriminatory ability stems from the first half of the feed-forward network performing a specific nonlinear transformation of the input data into a space in which the discrimination should be easier. A good discrimination between classes in the space of the hidden units is obtained by requiring a minimization of the output error.

Backpropagation networks with feedforward connections have by now been established as highly competent classifiers (Waibel *et al.*, 1989; Burr, 1988; Barnard & Casasent, 1989) but much remains to be discovered concerning the optimal design of such networks for particular applications. Issues such as the appropriate choice of features for input to the network Barnard *et al.*, 1991, the training methodology to be used (Jacobs, 1988) and the best network topology (Obradovic & Yan, 1990; Chester, 1990) has all been identified, but complete satisfactory solutions have not been offered for any of these problems.

The accuracy (and hence reliability) of neural net implementations can be improved through a systematic treatment of the two failure cases: training failures and test failures. The paper addresses the basic problem of improving MLP neural net implementations, i.e. increasing the generalization performance of the networks by developing methods for dealing with training and testing failure patterns.

One of the approaches to be explored is using the parameter LENGTH. In Launch Intercepor Problem 1, LENGTH is one of the parameters provided as the input

pattern. The parameter LENGTH acts as discriminant that will be used to identify whether a particular pattern is considered as the learnable *(easy to learn* or *ETL)* or *hard to learn* (HTL) pattern. The HTL pattern is then treated before training.

METHODOLOGY

The Launch Interceptor problem has been used in a number of software engineering experiments concerning correctness and reliability (see Knight & Leveson, 1986; Adams & Taha, 1992). Partridge and Sharkey (1994), and Yates and Partridge (1995) have applied the problem to neural networks. This is a well-defined, abstract problem that has been chosen as a case study for testing the relevant procedures since it offers a distinct advantage of supplying numerous training and test patterns with unambiguous outcomes. The problem involves an anti-missile system, which is used to classify radar images as indicative of a hostile missile, or not. The input for the system represents radar images (specified as a sequence of x,y coordinate points) together with 19 real-valued parameters and several small matrices which are used to control the interpretation of the radar images. The output is simply a decision *Launch* (when all 15 launch criteria are satisfied according to certain conditions) or *No-Launch* (when one or more of the 15 launch criteria is not satisfied by the input data). The various criteria upon which the decision depends are referred to as "launch interceptor conditions" (LIC's). LIC1, the acronym from Launch Interceptor Condition 1, is a *boolean* function that is *true* if the Euclidean distance between two points is greater than the value of another parameter LENGTH, and *false* otherwise. The points are given as two pairs of x and y coordinates (each in the interval [0,1] to six decimal places) LENGTH is as single value in the same interval to the same precision. Therefore LIC1 takes five input values i.e. $(x_1, y_1), (x_2, y_2)$, LENGTH and returns the value *true* or *false*.

To create training and test sets, LIDV the acronym from Launch Interceptor Data Vector is used to generate either rational or random patterns. Random sets are constructed by using a random number generator to construct each of the five parameters. For rational sets LIDV constructs the decision-boundary patterns by approximately setting the LENGTH parameter after randomly setting x_1, y_1, x_2 and y_2. LIDV determines the value of the LENGTH by calculating the actual dis-

tance between two coordinates [(x_1,y_1) and (x_2,y_2)] and adding or subtracting a certain percentage of this distance, a parameter whose bounds (i.e. minimum and maximum percentage distances) are determined by the user. For the purpose of the study, the lower bound for the rational training set is set to 0% and the upper bound is 20%, and for the rational test sets the upper bound is set to 100%. For the random patterns, no specification of the bound is required.

The main objectives of the experiments are firstly to determine whether the discriminant was a good predictor, and secondly to find out whether good prediction of *HTL/ETL* can be used to improve generalization performance. Having determined the *HTL/ETL* patterns, the next step is to perform experiments which should give some insight to the following plausible objectives:

1) To determine whether separating and modifying *HTL* patterns makes the patterns more learnable.
2) To determine whether learning improvement based on LENGTH as discriminant leads to better computational reliability (i.e. a reduction in the number of wrongly computed results).

Once the HTL patterns were identified by the discriminant, further treatments were applied to these patterns. For modification purposes, normalization methods were employed. The first normalization method is to sum the squares of each parameter (i.e. x_1, y_1, x_2, y_2 and LENGTH), take the square root of the sum, and then divide each element by the norm (Bigus, 1996; Cohn, 1994; Cohn, 1974). This method is known as the Euclidean norm. A second method of normalization is simply by dividing each parameter by the parameter that has the largest value for a particular pattern (Bigus, 1996). The second normalization method is referred to as the new normalization technique (new-norm).

The approach has been stimulated from the empirical studies conducted by Littlewood and Miller (1989) for software engineering research and Partridge and Sharkey (1992), Partridge (1990), Partridge and Griffith (1994), and Partridge and Yates (1995a,1995b) from neural networks experimental research. In conjunction with these studies, MLP networks (Rumelhart and McClelland,1986) with five input units, eight to 10 hidden units and one output unit (i.e. 5-8-1 to 5-10-1) are utilized in the experiments. Six training sets (three rational ones i.e. R1, R2 and R3, and three random ones T1, T2 and T3) are used to investigate the effect of normalizing on the training and test patterns. Each training set is composed of

1000 rational (or random) patterns trained using an online backpropagation algorithm with a learning rate of 0.05 and momentum of 0.5. Sarle (1995) suggested that sample size of 1000 is large enough that overfitting is not a concern, so the results for 1000 training cases will provide a standard comparison for generalization results from small sample sizes. The weight seed number is varied from 1, 2 and 3. Each MLP network is trained to convergence (i.e. every pattern learned to a tolerance of 0.5 or 50,000 epochs for all training sets) whichever is first.

The experiments were carried out in the same order as the objectives listed in the previous section. To improve the generalization for testing, the boundary patterns (i.e. rational training sets) were used. These patterns should have a higher probability of being a good separator of the input space. These boundary patterns are most likely to help the network generalize correctly (Sabutai & Tesauro, 1988; Huyser & Horowitz, 1988; Collins, 1994; Partridge & Collins, 1995). Hence, both rational and random training, and test sets are considered. In order to produce general results, ones with statistical validity, the component experiments were each repeated a number of times. The number of hidden unit is varied from eight to 10; training set from R1 to R3 (or from T1 to T3) and the weight seed from W1 to W3. Hence, the total number of networks generated was 27 (i.e. 3^3 that represents three weight seeds x three training sets x 3 hidden unit number) for random and another 27 for rational. The performance of the networks was tested on rational and random test patterns (each containing 10,000 test patterns).

DISCRIMINATION STRATEGIES

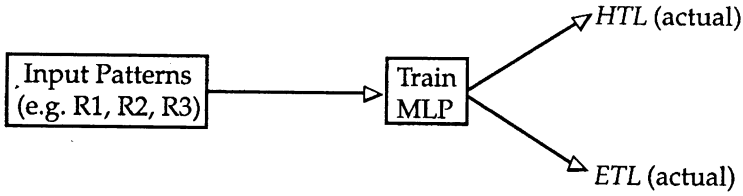
It is hypothesized that patterns with LENGTH $\leq$ 0.02 and the distance between two pairs of coordinates do not learn after training. Preliminary studies show that the unlearnt patterns lie close to the origin or along the boundary decision line. The experiment performed in this section is to determine whether LENGTH is a good discriminant in such a way that it is able to discriminate patterns that may be difficult to learn from the ones that are easy to learn.

Three rational (R1, R2 and R3) and three random (T1, T2 and T3) sets are used in the following experiments. These training sets will be trained as simple MLPs using different number of weight seeds (1, 2 and 3) and different number of hid-

den units (8, 9 and 10). As a result, a multiversion system with nine networks was obtained for each training set. Each pattern is considered as *HTL* if its parameter LENGTH $\leq$ 0.02, otherwise it will be considered as an *ETL* pattern. Having defined the *HTL* and *ETL* patterns, the evaluation of the discriminant is summarized in the following ways:

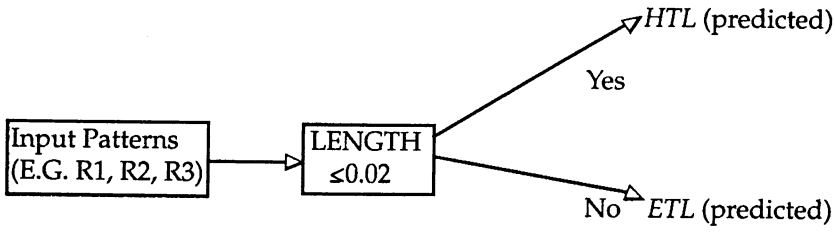
1. The results obtained by training R1, R2 and R3 (or T1, T2 and T3) are used as a baseline for comparison. In effect, the actual *HTL* and *ETL* patterns are identified. The procedure is illustrated in Figure 1.

Figure 1
The Procedure for Identifying the Actual HTL and ETL Patterns



2. The LENGTH is used as the discriminant to predict whether each input pattern of R1, R2 and R3 is HTL or ETL pattern. A pattern is considered as the predicted HTL if the parameter LENGTH of this pattern is less or equal to 0.02, otherwise it will be considered as the predicted *ETL*. The procedure is illustrated in Figure 2.

Figure 2
The Procedure for Identifying the Predicted *HTL* and *ETL* Patterns Using the LENGTH as the Discriminant



Having obtained the *HTL/ETL* patterns, several training and testing methods were explored, namely:

(1) The patterns are trained as MLPs without applying any treatments to the *HTL* patterns. This method is labelled as SR_{raw}.

(2) Split and Recombine (SR)
The *HTL* (or *HTC*) patterns are treated by normalization methods. Both *HTL* and *ETL* were recombined prior to training or testing. This method is labelled as SR_{norm} and $SR_{new\text{-}norm.}$

(3) Split and Separate (SS)
The *HTL* and *ETL* patterns are trained separately. The *HTL* are normalized before training or testing. This method is labelled as SS_{norm} and $SS_{new\text{-}norm.}$

(4) The *HTL* and *ETL* patterns are trained separately but the *HTL* patterns are not treated before training or testing. This method is labell

(5) ed as $SS_{raw.}$

The experiments will be carried out in the same order as the objectives listed in section 2.0.

RESULTS

The results exhibited in Table 1 indicate that the effect of SR training methods on the training performance is highly significant (p=0.00). At 5% significance level, the mean success of SR_{norm} and $SR_{new\text{-}norm}$ have significantly improved the training results of SR_{raw} ($p = 0.02$ and $p = 0.00$ respectively). When SS training methods were employed, the effect of these methods on the training is very significant (p=0.00). In particular, mean success of SS_{norm} and $SS_{new\text{-}norm}$ are very significant when compared to SR_{raw} ($p = 0.01$ and $p = 0.01$).

The unlearnt patterns which lie in the range of LENGTH between 0.0 and 0.02 (20 patterns) have become learnt when trained using other training methodologies (see also Table 2). Therefore, the training methodologies SR_{norm} and $SR_{new\text{-}norm}$ have made all unlearnt patterns whose LENGTH ≤ 0.02 and some patterns with LENGTH > 0.02 become learnt after training. The test results exhibited in Table 2 indicate that all SR techniques for rational patterns achieve at least 0.38% higher

Table 1
Tha Training and Test Results for Rational Patterns Using LENGTH as the Discriminant

	Training (in %)			Testing (in %)			
Method	R4	R5	R6	Av.	Min	Max	Average
SR_{raw}	98.53	97.86	97.08	97.82	83.28	99.26	98.29
SR_{norm}	98.86	99.23	98.83	98.97	98.60	99.35	98.99
$SR_{new\text{-}norm}$	99.43	99.52	99.43	99.46	99.35	98.59	99.07
SS_{norm}	98.96	99.29	99.20	99.15	93.53	99.19	97.16
$Ss_{new\text{-}norm}$	98.96	99.29	99.20	99.15	93.41	98.14	95.87
SS_{raw}	98.96	98.29	99.19	98.68	94.70	99.06	96.97

Table 2
The Rational Patterns that Do Not Learn After Training

	No. of patterns whose LENGTH $\leq$ 0.02				
Training Methodology	R1	R2	R3	The patterns whose LENGTH $\leq$ 0.02 that do not learn after training	Total Unique Unlearnt
SR_{raw}	78	68	71	20	234
SR_{norm}	26	25	28	0	184
$SR_{new\text{-}norm}$	25	25	27	0	166

than SR_{raw}. The effects of the weight seed numbers, number of hidden units, training sets and the testing methods on the generalization performance are not significant ($p = 0.25$, $p = 0.51$, $p = 0.547$, $p = 0.236$) at 5% significance level. The only SR method that has significantly improved the generalization performance at 10% significance level is the $SR_{new\text{-}norm}$ ($p = 0.094$).

After testing, all techniques have decreased the number of pattern failures i.e. 10.50% by rescaling, 44.59% by normalizing and 51.77% by new normalizing techniques. Highly significant differences were found in mean test failures between testing methodologies when rational test sets were tested on rational weights ($p<0.0001$). Furthermore, in both normalizing techniques none of the patterns whose LENGTH ≤ 0.02 failed after testing (see Table 3). On the other hand, when the SS methods were employed the generalization of the network decrease. This may be due to two possibilities, namely the overtraining problem, and the way the patterns were discriminated into *HTC* and *ETC* subsets. These factors will be considered further by employing different discriminant procedures in further experiments.

For random patterns, the effect of SR training methods is not significant. None of the patterns whose LENGTH ≤ 0.02 did not learn after training. The distribution of unlearnt patterns lie close to the decision boundary line especially between $0.04 \leq$ LENGTH ≤ 0.08. After rescaling, most of the unlearnt patterns after training lie between $0.04 \leq$ LENGTH ≤ 0.06. The normalizing technique, however, introduced more unlearnt patterns at LENGTH between 0.04 and 0.08. Using the new normalizing technique, most of the unlearnt patterns at LENGTH between 0.04 and 0.08 learnt after training. Hence, the rescaling and new normalizing techniques have corrected some failed patterns whose value of LENGTH > 0.02.

The total number of SR_{raw} pattern whose LENGTH ≤ 0.02 in random training sets T1, T2, T3 are 27, 13 and 11 respectively, but these patterns learnt after training. After treatments, two patterns were affected by normalizing and nine by new normalizing techniques. One interesting point to note is that for all training methodologies, all patterns whose LENGTH ≤ 0.02 for training sets T1, T2 and T3 learnt after training. This implies that all unlearnt patterns after training lie in the range of $0.02 >$ LENGTH ≤ 1.0. We perceived that the treatments make the unlearnt pattern whose LENGTH ≤ 0.02 become learnt after training, however in the case of random training, the treatments using $SR_{new\text{-}norm}$ have reduced the number of

Table 3
The Number of Failures After Testing Using Rational Patterns

Testing Methodology	No. of patterns whose LENGTH ≤ 0.02	Patterns whose LENGTH ≤ 0.02 that fail after testing	Total failures
SR_{raw}	5095	353	933
SR_{norm}	2452	0	517
$SR_{new\text{-}norm}$	2441	0	450

Table 4
The Training and Test Results for Random Patterns Using LENGTH as Discriminant

	Training (in %)				Testing (in %)		
Method	T4	T5	T6	Av.	Min	Max	Average
SR_{raw}	97.96	99.92	99.81	99.24	83.28	99.26	96.86
SR_{norm}	99.96	99.98	99.81	99.91	95.73	98.30	97.57
$SR_{new\text{-}norm}$	99.81	99.81	99.81	99.81	96.13	97.86	97.16
SS_{norm}	99.90	99.93	99.68	99.84	95.76	98.40	97.52
$SS_{new\text{-}norm}$	99.90	99.93	99.68	99.84	95.76	98.40	97.52
SS_{raw}	99.90	99.93	99.68	99.84	95.76	98.40	97.52

Table 5
The Number of Failures After Testing Using Random Patterns

Testing Methodology	No. of patterns whose LENGTH ≤ 0.02	Patterns whose LENGTH ≤ 0.02 that fail after testing	Total failures
SR_{raw}	194	1	1432
SR_{norm}	185	0	1452
$SR_{new\text{-}norm}$	159	1	1313

unlearnt patterns whose values of LENGTH > 0.02. The results presented in Table 4 show that the effects of weight seed numbers, the number of hidden units, the training sets and the testing methods on the generalization performance of random patterns are not significant. All other SR and SS techniques achieved at least 0.30% higher than SR_{raw}. However, all of these methods have not achieved a significant improvement on the generalization of networks at 5% significance level.

In random testing, the total number of raw patterns whose LENGTH $\leq$ 0.02 is 194. Only one out of these patterns failed after testing. After applying normalizing and new normalizing techniques, the total number of modified patterns with LENGTH $\leq$ 0.02 has reduced (see Table 5). The total number of failures has also decreased by 8.31% using new normalizing technique. However, SR_{norm} does not decrease the number of unique failure after testing. When random test sets were tested on random weights the mean test failures between each methodology was not significant at all ($p < 0.618$). Nevertheless, the mean test failures for SR_{raw} (mean = 306.11) has reduced to 298.56 when treated with $SR_{new\text{-}norm.}$

CONCLUSION

The results presented in the previous sections indicate that normalizing and new normalizing techniques have improved the average generalization of the networks which consists of rational patterns tested on rational weights. For random patterns, only normalizing technique shows a slight decrease in the average performance. Since the testing methodologies introduced in the experiments show some positive results, a more rigorous statistical tool could be introduced to discriminate the patterns to be modified in both training and test sets. One possible approach is to introduce the use of discrimination analysis technique (Hand, 1981; Ripley, 1993; Bishop, 1995; Krzanowski & Marriot, 1995) or MLP networks trained to perform the discrimination (Webb & Lowe, 1990). These two techniques will be explored in future experiments.

REFERENCES

Adams, J. & Taha, A. (1992). An experiment in software redundancy with diverse methodologies. *Proceedings of the 25th Hawaii International Conference on System Sciences*, 83-90.

Barnard, E. & Casasent, D. (1989). A comparison between criterion functions for linear classifiers, with an application to neural nets. *IEEE Trans. Syst., Man, Cybern.*, 19 (Sept./Oct), 1030-1041.

Barnard, E., Cole, R., Vea, M., & Alleva, F. (1991). Pitch detection with a neural-net classifier. *IEEE Trans. Signal Processing*, 39 (Feb), 298-307.

Bigus, J. P. (1996). *Data Mining with Neural Networks.* New York: Mc-Graw Hill.

Bishop, C. (1995). *Neural Networks for Pattern Recognitions.* Oxford: Clarendon Press.

Burr, D. (1988). Experiments on neural net recognition of the spoken and written text. *IEEE Trans. Acoust, Speech, Signal Processing*, ASSP-36, July, 1162-1168.

Chester, D. (1990). Why two hidden layers are better than one. *Proceedings Int. Joint Conference Neural Networks*, Jan, I-265-I-268, Washington, DC.

Cohn, P. (1974). *Algebra* (Volume 1). London: John Wiley and Sons.

Cohn, P. (1994). *Elements of Linear Algebra.* London: Chapman and Hall.

Collins, T. (1994). *An investigation into neural network training.* Master's thesis, Computer Science Dept., Exeter University.

Huyser, K, & Horowitz, M. (1988). Generalization in connectionist networks that realize boolean functions. In Touretzky, F. Hinton, G., & Sejnowski, T. (eds.), *Proceedings of the 1988 Connectionist Models Summer School* (pp. 191-200). Morgan Kauffmann Publishers, INC.

Hand, D. (1981). *Discrimination and Classification.* London: John Wiley and Sons.

Jacobs, R. (1988). Increased rates of convergence through learning rate adaptation. *Neural Networks*, 1(1), 295-308.

Knight, J. & Leveson, N. (1986). An experimental evaluation of the assumption of independence in multiversion programming. *IEEE Trans. Software Eng.*, 12(1), 96-109.

Krzanowski,W. &Marriot, F. (1995). *MultivariateAnalysis*(Part2).England:Arnold.

Littlewood, B. & Miller, D. (1989). Conceptual modeling of coincident failures in multiversion software. *IEEE Transactions on Software Engineering*, 15(12).

Obradovic, Z. & Yan, P. (1990). Small depth polynomial size neural networks. *Neural Computation*, 2 (Winter), 402-404.

Partridge, D. (1990). Network generalization differences quantified. *Neural Networks*, 9 (2),263-271.

Partridge, D. & Collins, T. (1995). Neural net training: Random versus systematic. In Taylor, J. (eds.), *Neural Networks* (pp. 85-93). United Kingdom: Alfred Waller Limited.

Partridge, D. & Griffith, N. (1994). *Strategies for improving neural net generalization* (Technical Report 301). Exeter: Department of Computer Science, Exeter University.

Partridge, D. & Sharkey, N. (1992). *Neural networks as a software engineering technology* (Technical Report 237). Exeter: Department of Computer Science, Exeter University.

Partridge, D. & Sharkey, N. (1994). *Neural computing for software reliability* (Technical Report 292). Exeter: Department of Computer Science, Exeter University.

Partridge, D. & Yates, W. (1995a). *Letter recognition using neural networks: A comparative study* (Technical Report 334). Exeter: Department of Computer Science, Exeter University.

Partridge, D. & Yates, W. (1995b). *Engineering mutliversion neural-net systems* (Technical Report 320). Department of Computer Science, Exeter university.

Ripley, B. (1993). Statistical aspecets of neural networks. In Nilesen, O., Jensen, J., N. Kendall, W. (eds.), *Network & Chaos: Statistical & Probabilistic Astects* (pp.41-123). United Kingdom: Chapman and Hall.

Rumelhart, D. & McClelland, J. (1986). Learning internal representations by error propagation. In Rumelhart, D., Hinton, G., & Williams, R. (eds.). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* (Volume 1, pp. 318-362). MIT Press: Bradford Books.

Sabutai, A. & Tesauro, G. (1988). Scaling and generalization in neural networks. In Touretzky, D., Hinton, G., & Sejnowski, T. (eds.). *Proceedings of the 1988 Connectionist Models Summer School* (pp. 3-10). Morgan Kauffmann Pulishers, INC.

Sarle, W. (1995). Stopped training and other remedies for overfitting. *In Proceedings of the 27th Symposium on the Interface.*

Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., & Lang, K. (1989). Phoneme recognition using time-delay neural networks. *IEEE Trans. Acoust, Speech, Signal Processing*, 37, 328-339.

Webb, A. R. & Lowe, D. (1990). The optimised internal representation of multilayer classifier networks performs nonlinear discriminant analysis. *Neural Networks*, 3, 367-375.

Yates, W. & Partridge, D. (1995). *Use of methodological diversity to improve neural net generalization* (Technical Report 316). Department of Computer Science, Exeter University.