A Cascade Network Algorithm Employing Progressive RPROP

N.K. Treadgold and T.D. Gedeon
School of Computer Science & Engineering
The University of New South Wales
Sydney N.S.W. 2052 AUSTRALIA
{ nickt | tom }@cse.unsw.edu.au

ABSTRACT

Cascade Correlation (Cascor) has proved to be a powerful method for training neural networks. Cascor, however, has been shown not to generalise well on regression and some classification problems. A new Cascade network algorithm employing Progressive RPROP (Casper), is proposed. Casper, like Cascor, is a constructive learning algorithm which builds cascade networks. Instead of using weight freezing and a correlation measure to install new neurons, however, Casper uses a variation of RPROP to train the whole network. Casper is shown to produce more compact networks, which generalise better than Cascor.

INTRODUCTION

The Cascade Correlation algorithm (Fahlman and Lebiere, 1990) is a very powerful method for training artificial neural networks. Cascor is a constructive algorithm which begins training with a single input layer connected directly to the output layer. Neurons are added one at time to the network and are connected to all previous hidden and input neurons, producing a cascade network. When a new neuron is to be added to the network, all previous network weights are 'frozen'. The input weights of the neuron which is about to be added are then trained to maximise the correlation between this neuron's output and the remaining network error. The new neuron is then inserted into the network, and all weights connected to the output neurons are then trained to minimise the error function. Thus there are two training phases: the training of hidden neuron weights, and the training of output weights. Both training phases use the QuickProp algorithm (Fahlman, 1988).

The separation of training phases allows Cascor to train a pool of neurons (with different random starting weights), and to select the neuron with the best resulting correlation for insertion. The training of both hidden neurons and the output weights is continued until the error stops decreasing by a set amount. The freezing of weights results in a very efficient algorithm, since it allows network values to be cached (and hence they need not be recalculated when a new neuron is added). In addition there is no back propagation of errors since only one layer of weights is trained at a given time.

While Cascor has been shown to be very successful (Fahlman, 1990), two drawbacks have been pointed out. First, Kwok and Yeung (1993) have shown that the technique of weight freezing can result in excessively large networks. They reason that weight freezing can result in early hidden units which are poor feature detectors. The network then requires further hidden units to fix the errors introduced by these earlier units. Second, it has been demonstrated that Cascor does not generalise well on regression and some classification problems (Hwang, You, Lay and Jou, 1996; Adams and Waugh, 1995). Hwang et al. explain these results by pointing out that the use of the

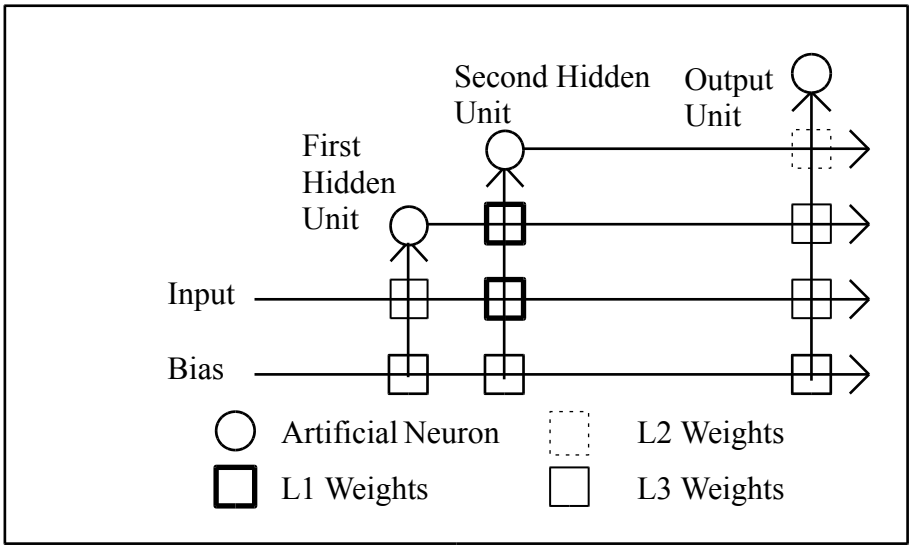
correlation measure in Cascor forces the hidden units to saturate, which produces jagged edges in the network outputs.

THE CASPER ALGORITHM

Casper uses a modified version of the RPROP algorithm (Riedmiller and Braun, 1993; Riedmiller, 1994) for network training. RPROP is a gradient descent algorithm which uses separate adaptive learning rates for each weight. Each weight begins with an initial learning rate, which is then adapted depending on the sign of the error gradient seen by the weight as it traverses the error surface.

The Casper algorithm constructs cascade networks in a similar manner to Cascor: Casper starts with a single hidden neuron and successively adds single hidden neurons. RPROP is used to train the whole network each time a hidden neuron is added. RPROP is modified, however, such that when a new neuron is added the initial learning rates for the weights in the network are reset to different values, depending on the position of the weight in the network. The network is divided into three separate regions, each with its own initial learning rate L1, L2 and L3 (Figure 1). The first region is made up of all weights connecting to the new neuron from previous hidden and input neurons. The second consists of all weights connecting the output of the new neuron to the output neurons. The third is made up of the remaining weights, which consist of all weights connected to, and coming from, the old hidden and input neurons.

Figure 1: The Casper architecture - a second hidden unit has just been added. The vertical lines sum their inputs.



The values of L1, L2 and L3 are set such that L1 >> L2 > L3. The reason for these settings is similar to the reason that Cascor uses the correlation measure: the high value of L1 as compared to L2 and L3 allows the new hidden neuron to learn the

remaining network error. Similarly, having L2 larger than L3 allows the new neuron to reduce the network error, without too much interference from other weights. This interference has been termed the 'herd effect' (Fahlman, 1990).

Importantly, however, no neurons are frozen, and hence if benefit can be gained by the network by modifying an old weight, this occurs, albeit at an initially slower rate than the weights connected to the new unit. Thus Casper retains the benefits of the weight freezing and the correlation techniques of Cascor, while removing both the saturation problems caused by the correlation measure, and the permanent installation of poorly performing neurons caused by weight freezing.

Casper also makes use of weight decay as a means to improve the generalisation properties of the constructed network. After some experimentation we found that the addition of a Simulated Annealing (SA) term applied to the weight decay, as used in the SARPROP algorithm (Treadgold and Gedeon, 1996) often improved convergence and generalisation. Each time a new hidden unit is inserted, the weight decay begins with a large magnitude, which is then reduced by the SA term. The amount of weight decay is proportional to the weight magnitude squared, which results in larger weights being decayed more rapidly. The error gradient used in Casper thus becomes:

$$\delta E/\delta w_{ij} = \delta E/\delta w_{ij} - k^* \operatorname{sign}(w_{ij})^* w_{ij}^2 * 2^{-0.01^*HEpoch}$$

HEpoch in the above formula refers to the number of epochs elapsed since the addition of the last hidden neuron, *sign* returns the sign (positive/negative) of its operand, and *k* is a user defined parameter which effects the magnitude of weight decay used.

In Casper a new neuron is installed after the decrease of the RMS error has fallen below a set amount. The RMS error must fall by at least 1% of its previous value in a given time period. The time period over which this measure is taken is given by the formula: 15+P*N, where N is the number of currently installed neurons, and P is a parameter set prior to training. This formula was found experimentally to give the best overall convergence properties. The result of this training method is that Casper increases the period over which the network is trained as the network grows in size.

BENCHMARKING RESULTS

To test the effectiveness of the Casper algorithm, we decided to compare its performance against that of the Cascor algorithm on a number of benchmark problems. In Casper a number of parameters require setting. The following (standard) parameter values were used for RPROP: $\eta^+ = 1.2$, $\eta^- = 0.5$, $\Delta_{max} = 50$, $\Delta_{min} = 1x10^{-6}$. In addition, a constant value of 0.0001 was added to the derivative of the sigmoid in order to overcome the 'flat spot' problem (Fahlman, 1988). The hyperbolic arctan error function (Fahlman, 1988) was used for classification problems, while the standard sum of squares error function was used for regression problems. All weights were initialised to random values in the range -0.7 to 0.7, and the standard symmetric sigmoid non linearity (-0.5, 0.5) was used for the hidden units. Training of the initial network used the initial update value $\Delta_0 = 0.2$. The values of L1, L2, and L3 were set

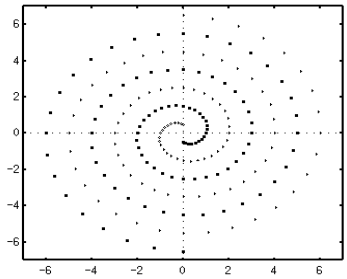
to 0.2, 0.005, and 0.001 respectively. All these parameter values were found to be problem independent, and hence were treated as constants. The remaining parameter values, k (the weight decay value) and N (the training length), were set depending on the problem.

The Cascor algorithm used for benchmarking was obtained from the public domain Carnegie Mellon University (CMU) AI Repository. For all comparisons, a pool of eight candidate neurons were used and a maximum learning iteration of 100 was set for both the hidden and output neurons, as used by Fahlman (1990). All further Cascor parameters were kept at the default values.

Two Spirals Benchmark

The two spiral benchmark consists of two interlocked spirals, each made up of 97 points (Figure 2). The network must learn to distinguish the two spirals. This problem was used by Fahlman (1990) to demonstrate the effectiveness of the Cascor algorithm on a problem known to be very difficult for traditional Back Propagation to solve (Fahlman, 1990). In order to compare Casper and Cascor on this problem, 100 independent runs were performed using each algorithm. The standard test set for the two spirals dataset (as supplied with the Cascor algorithm) was used to measure the resulting generalisation ability of the networks. This test set consists of two spirals each made up of 96 points, slightly rotated relative to the original spirals. The parameter values used for the Casper algorithm were P=5 and k=0.005. Training was halted when all network outputs were within 0.2 of the required training outputs.

Figure 2: The two spirals training set



The average, standard deviation and median for the following characteristics were measured: epochs trained, hidden units installed, number of connection crossings and percentage correct on the test set. Fahlman (1990) defines the term connection crossings as “the number of multiply-accumulate steps to propagate activation values forward through the network and error values backward”. This term is a more valid way to compare learning times than number of epochs trained, since Cascor makes use of weight freezing and caching (Fahlman, 1990) which greatly improves the algorithm’s efficiency.

The results for Casper and Cascor on the two spirals problem are shown in Tables 1 and 2 respectively. Figures 3 is a histogram of the number of hidden units created by the Casper and Cascor algorithms over the 100 training runs. Figures 4, 5 and 6 are plots of the best results obtained (in terms of success on the test set) from the Casper

runs (produced by networks of hidden unit size 12, 12, and 14 respectively), and Figures 7, 8, and 9 are plots of the best results obtained from the Cascor runs (produced by networks of hidden unit size 14, 17 and 18 respectively).

Table 1: Two spiral results for Casper

	Epochs	Hidden Units	Con. Crossings	Test Set %
Average	2437	13.49	9.57×10^7	97.80
Std. Dev.	627	2.41	5.37×10^7	2.22
Median	2307	13.0	8.56×10^7	98.44

Table 2: Two spiral results for Cascor

	Epochs	Hidden Units	Con. Crossings	Test Set %
Average	1686	15.96	2.02×10^7	96.13
Std. Dev.	209	2.17	4.47×10^6	2.11
Median	1689	16.0	1.99×10^7	96.35

Figure 3: Hidden unit histogram for the two spiral problem

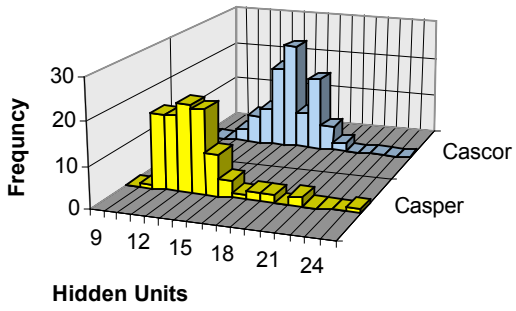


Figure 4: Casper 12 hidden **Figure 5:** Casper 12 hidden **Figure 6:** Casper 14 hidden

Fig 7: Cascor 14 hidden **Fig 8: Cascor 17 hidden** **Fig 9: Cascor 18 hidden**



Regression Benchmark

Five regression functions were chosen to compare Casper and Cascor. The functions are described in detail in Hwang et al. (1994), and are shown below:

- Simple interaction function

$$f^{(1)}(x_1, x_2) = 10.391((x_1 - 0.4)(x_2 - 0.6) + 0.36).$$

- Radial function:

$$f^{(2)}(x_1, x_2) = 24.234(r^2(0.75 - r^2)),$$
$$r^2 = (x_1 - 0.5)^2 + (x_2 - 0.5)^2$$

- Harmonic function:

let $y_1 = x_1 - 0.5$ and $y_2 = x_2 - 0.5$

$$f^{(3)}(x_1, x_2) = 42.659(0.1 + y_1(0.05 + y_1^4 - 10 y_1^2 y_2^2 + 5y_2^4)).$$

- Additive function:

$$f^{(4)}(x_1, x_2) = 1.3356(1.5(1 - x_1 + e^{2x_1 - 1}\sin(3p(x_1 - 0.6)^2) + e^{3(x_2 - 0.5)}\sin(4p(x_2 - 0.9)^2)$$

- Complex interaction function:

$$f^{(5)}(x_1, x_2) = 1.9(1.35 + e^{x_1}\sin(13(x_1 - 0.6)^2)e^{-x_2}\sin(7x_2)).$$

The set up of training and test data follows the method of Hwang et al. (1994). For each function two sets of training data were created, one noiseless and one noisy, created using 225 randomly selected pairs [0,1] of abscissa values $\{(x_1,x_2)\}$. The same abscissa values were used for all five functions. The noisy data was created by adding independent and identically distributed (*iid*) Gaussian noise, with zero mean and unit variance, giving an approximate signal to noise ratio of 4 (Hwang et al., 1994). For each function an independent test set of size 2500 was generated on a regularly spaced grid $[0,1]^2$, as used by Hwang et al. (1996).

The network parameters for Cascor were kept the same as for the two spirals runs, except that a linear output node was used, as is standard for regression problems.

Casper also employed a linear output node, and the values for N and k were set to 7 and 0.0005 respectively.

The fraction of variance unexplained (FVU) was the measure chosen to compare the performance of Casper and Cascor on the test set (Hwang et al. 1994). FVU is defined as:

$$FVU = \frac{\sum_{l=1}^{N} (\tilde{f}(x_l) - f(x_l))^2}{\sum_{l=1}^{N} (f(x_l) - \bar{f}(x_l))^2}$$

For each regression function 100 runs were performed using different random starting weight values. Training was continued for both algorithms until 30 hidden units had been installed. The FVU on the test set was measured after the installation of each hidden unit for both Casper and Cascor. The median FVU values after each hidden unit had been installed is shown in Figures 10 to 14.

DISCUSSION

The results on the two spirals problem demonstrates that Casper is able to produce networks which are more compact in terms of the number of hidden units than Cascor, even though Cascor trains a pool of eight hidden neurons and selects the best. The difference in the median network size produced by the two algorithms is 3 hidden units, which because of the cascade network structure equates to Casper using 54 less weights than Cascor in the median case. Cascor, however, is less expensive in terms of connection crossings than Casper. This result is mainly due to the fact that Cascor is able to cache calculations since it uses weight freezing, while Casper still employs back propagation of errors.

Figure 10: Simple interaction function

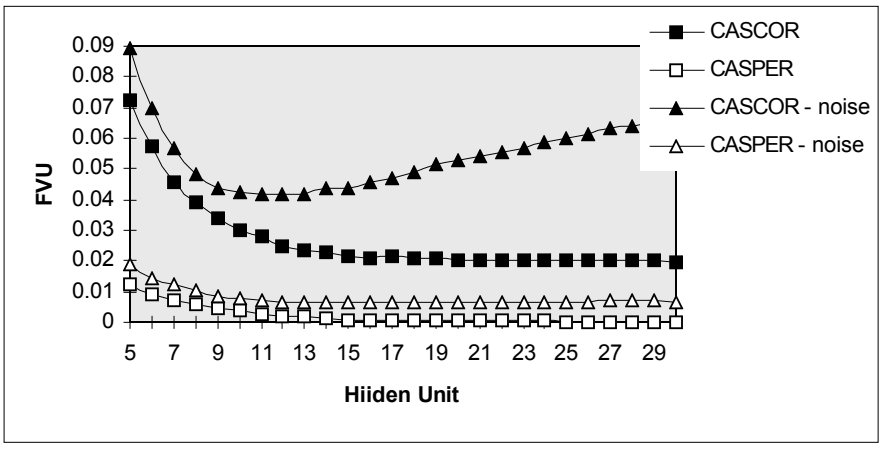


Figure 11: Radial function

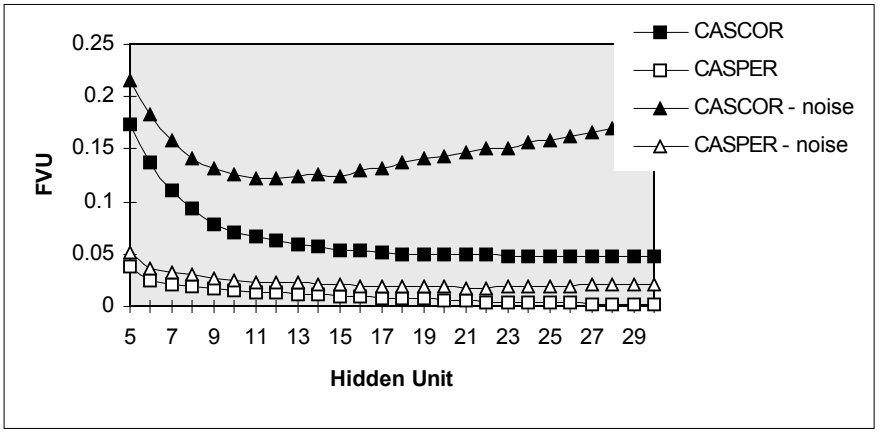
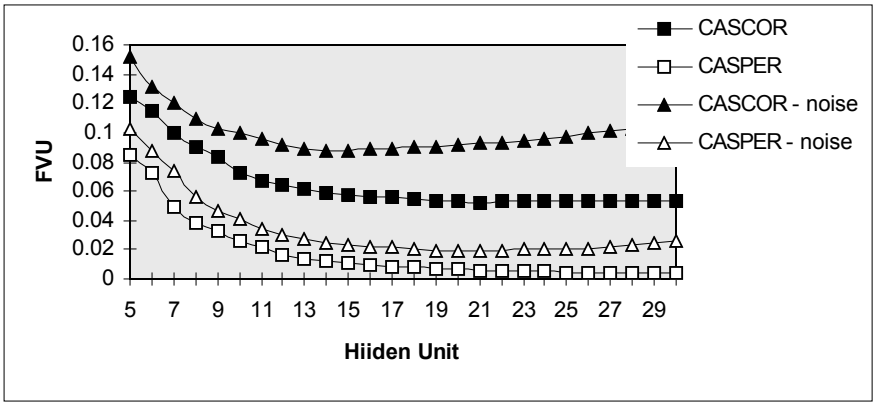


Figure 12: Harmonic function



For the two spirals dataset, out of the 100 training runs, Casper produced 30 networks which gave 100% correct results on the test set, while Cascor produced only 3 such networks. The outputs of these 3 networks are shown in Figures 8 to 10, in which the

characteristic jagged outputs produced by Cascor can be seen. On the other hand, Figures 5 to 7 produced by Casper, demonstrate the smooth outputs produced by this algorithm, and illustrate qualitatively Casper's better generalisation even in the cases where 100% correct classification on the test set was obtained by both algorithms. As expected, the Cascor algorithm performs poorly on regression datasets, as predicted by Hwang et al (1996). Casper, on the other hand, is able to reliably produce good solutions on these datasets. In particular, Cascor can be seen to suffer from over fitting on the noisy datasets as seen in Figure 11, while Casper succeeds in maintaining a good FVU as training continues.

Figure 13: Additive function

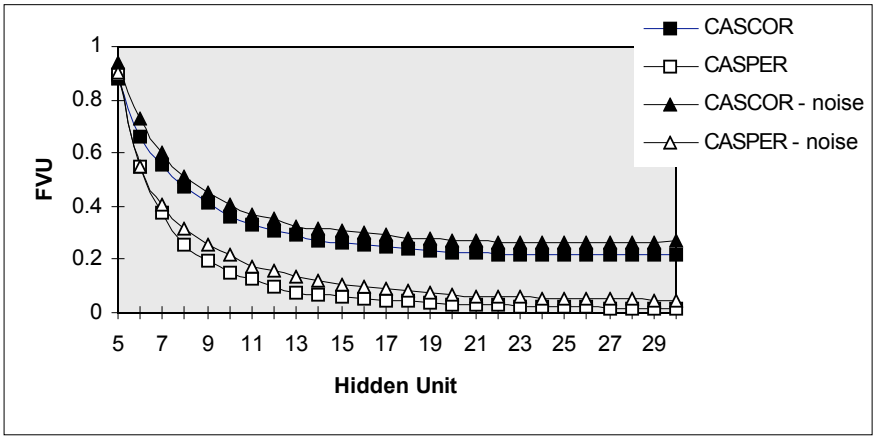
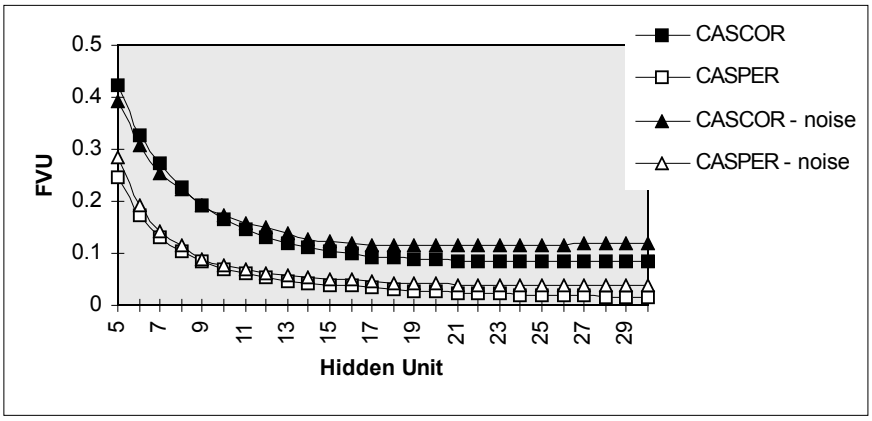


Figure 14: Complex Interaction function



The weight decay term employed by the Casper algorithm was found to effect both the resulting generalisation ability of the networks, as well as the number of hidden units installed. In general, the less weight decay used (obtained from smaller k values), the more compact the final network (and the shorter the training time); however the network generalisation usually suffered. The use of the SA term in the weight decay was found to produce more compact networks, while still maintaining good generalisation characteristics. We believe that this occurs because the high weight decay values after the addition of a new hidden unit act as a noise factor, allowing the network to access more promising parts of the error surface, which results in a more compact network. The high initial weight decay also results in reduced weight magnitudes, and thus the hidden neurons are less likely to saturate and produce jagged outputs.

The results of these comparisons highlight the fact that the correlation and weight freezing techniques used by Cascor are not necessary to solve complex problems. It is the cascade connections, however, which allow the hidden units to act as higher order feature detectors and thus gives both Cascor and Casper the ability to solve such problems. The use of 'higher order' neurons has been utilised successfully in other training algorithms such as Projection Pursuit Learning methods (Hwang et al, 1994).

CONCLUSION

A new constructive algorithm, Casper, is proposed. We have shown that weight freezing and the correlation measure employed in Cascor are not required to produce networks capable of solving complex problems. Casper is shown to produce networks which are both more compact in terms of hidden neurons and generalise better than those produced by the Cascor algorithm.

REFERENCES

Adams, A., and Waugh, S. (1995) Function Evaluation and the Cascade-Correlation architecture In *Proc. 1995 IEEE Int. Conf. Neural Networks.* pp. 942-946.

Fahlman, S.E. (1988) Faster learning variations on backpropagation: An empirical study. In *Proc. 1988 Connectionist Models Summer School.* San Mateo, CA: Morgan Kauffman

Fahlman, S.E., and Lebiere, C. (1990) The cascade-correlation learning architecture. In *Advances in Neural Information Processing II*, Touretzky, Ed. San Mateo, CA: Morgan Kauffman, 1990, pp. 524-532.

Hwang, J., Lay, S., Maechler, R. And Martin, D. (1994) Regression Modeling in Back-Propagation and Projection Pursuit Learning. *IEEE Trans. Neural Networks* vol. 5, no. 3. pp. 342-353.

Hwang, J., You, S., Lay, S., and Jou, I. (1996) The Cascade-Correlation Learning: A Projection Pursuit Learning Perspective. *IEEE Trans. Neural Networks* vol. 7, no. 2. pp. 278-289.

Kwok, T., and Yeung, D. (1993) Experimental Analysis of Input Weight Freezing in Constructive Neural Networks. In *Proc. 1993 IEEE Int. Conf. Neural Networks.* pp. 511-516.

Riedmiller, M. and Braun, H. (1993) A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm. In: Ruspini, H., (Ed.) *Proc. of the ICNN 93, San Francisco*, pp. 586-591.

Riedmiller, M. (1994) Rprop - Description and Implementation Details, Technical Report, University of Karlsruhe.

Treadgold, N.K., and Gedeon, T.D. (1996) A Simulated Annealing Enhancement to Resilient Backpropagation. *Proc. Int. Panel Conf. Soft and Intelligent Computing, Budapest* pp. 293-298.