Extending and Benchmarking the CasPer Algorithm

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

Abstract - The CasPer algorithm is a constructive neural network algorithm. CasPer creates cascade network architectures in a similar manner to Cascade Correlation. CasPer, however, uses a modified form of the RPROP algorithm, termed Progressive RPROP, to train the whole network after the addition of each new hidden neuron. Previous work with CasPer has shown that it builds networks which generalise better than CasCor, often using less hidden neurons. This work adds two extensions to CasPer. First, an enhancement to the RPROP algorithm, SARPROP, is used to train newly installed hidden neurons. The second extension involves the use of a pool of hidden neurons, each trained using SARPROP, with the best performing selected for insertion into the network. These extensions are shown to result in CasPer producing more compact networks which often generalise better than those produced by the original CasPer algorithm.

Keywords - Neural, Network, Constructive, Cascade, RPROP.

1 INTRODUCTION

The CasPer [1] algorithm has been shown to be a powerful method for training neural networks. CasPer is a constructive algorithm which inserts hidden neurons one at a time to form a cascade architecture, similar to Cascade Correlation (CasCor) [2]. CasPer has been shown to produce networks with fewer hidden neurons than CasCor, while also improving the resulting network generalisation, especially with regression tasks [1]. The reasons for CasPer's improved performance is that it does not use either Cascor's correlation measure, which can cause poor generalisation performance [3], or weight freezing, which can lead to oversize networks [4].

A difficult problem faced by both CasPer and CasCor is that the newly created hidden neuron may have difficulty in converging to a good solution on the error surface. One main cause for this poor convergence may be the presence of local minima. CasCor addresses this problem by creating a pool of hidden neurons, each with a different set of starting weights, thus enabling a wider search of the error surface and reducing the chance of convergence to a poor local minimum.

In order to improve the convergence ability of CasPer, two extensions are proposed. The first is to employ the SARPROP algorithm [5] to train the newly inserted hidden neuron. SARPROP is based on the RPROP algorithm [6], and uses Simulated Annealing to enhance the convergence properties of RPROP. SARPROP has been shown to be successful in escaping local minima [5], a property which will enable a better search of the error surface by the new hidden neuron. The second extension

involves CasPer training a pool of hidden neurons, as is done in CasCor. Each hidden neuron in the pool is trained using SARPROP, as in the first extension.

2 THE CASPER ALGORITHM

CasPer uses a modified version of the RPROP algorithm 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. This results in the update value for each weight adaptively growing or shrinking as a result of the sign of the gradient seen by that weight.

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

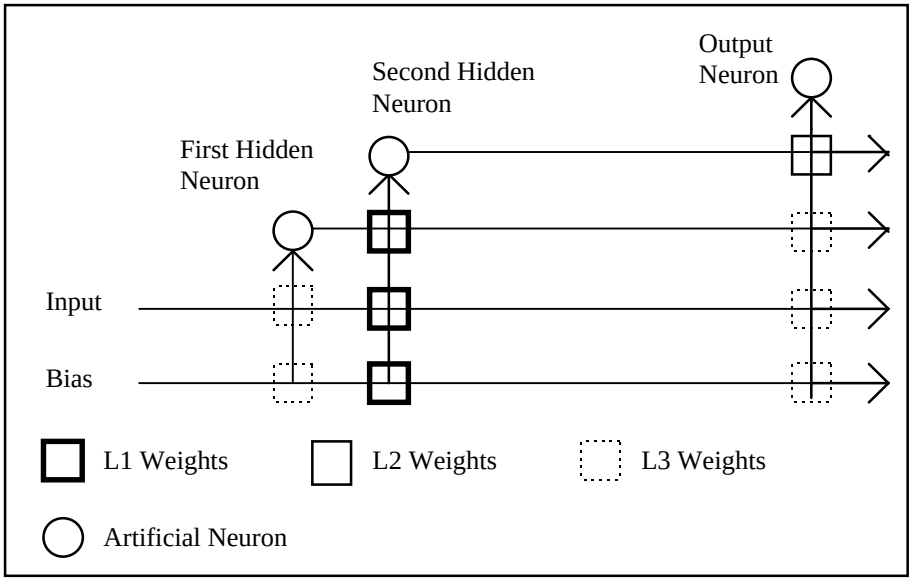

Fig. 1. The CasPer architecture - a second hidden neuron has just been added. The vertical lines sum all incoming inputs.

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. Importantly, however, no weights 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 neuron.

CasPer also makes use of weight decay as a means to improve the generalisation properties of the constructed network. After some experimentation it was found that the addition of a Simulated Annealing (SA) term applied to the weight decay, as used in the SARPROP algorithm [5], often improved convergence and generalisation. Each time a new hidden neuron 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 square of the weight magnitude. This results in larger weights decaying more rapidly. The error gradient used in CasPer thus becomes:

$$\delta E/\delta w_{ij}^{CasPer} = \delta E/\delta w_{ij} - D^* \text{sign}(w_{ij})^* w_{ij}^2 * 2^{-0.01*HEpoch}$$

HEpoch 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 *D* is a user defined parameter which effects the magnitude of weight decay used.

In CasPer a new neuron is inserted when the RMS Error falls by less than 1% of its previous value. The time period in epochs over which this measure is taken is given by the heuristic formula: 15 + L*N, which was found experimentally to give good results (*N* is the number of currently inserted neurons, and *L* is the training length parameter which is set prior to training). The result of this training method is that CasPer increases the period over which the network is trained as the network grows in size.

3 EXTENSIONS TO THE CASPER ALGORITHM

The first extension to the CasPer algorithm involves the use of the SARPROP algorithm to train all the incoming weights of the newly inserted hidden neuron (that is, all weights in group L1). All remaining weights are still trained by RPROP.

The SARPROP algorithm is based on RPROP, but uses a noise factor to enhance the ability of the network to escape from local minima. Noise is added to the update value of a weight when both the error gradient changes sign in successive epochs (indicating the presence of a minimum), and the magnitude of the update value is less than a threshold value. The amount of noise added falls as training continues via a Simulated Annealing term. This combination of techniques has been shown to improve RPROP's ability to escape local minima [5]. The equation used to modify the update value, Δ_{ij}, whenever a change in error gradient sign is encountered is:

$$\begin{aligned} \Delta_{ij}(t+1) &= \Delta_{ij}(t)\,\eta^- + k_1 rS \quad \text{if } \Delta_{ij}(t) < k_2 * S \\ &= \Delta_{ij}(t)\,\eta^- \quad \text{otherwise} \end{aligned}$$

where r is a random number between 0 and 1, η^- is the standard RPROP update constant, k_1 and k_2 are constants, and S is the SA term which is set to $2^{-0.03*HEpoch}$. The version of CasPer employing this extension will be termed S_CasPer.

The second extension involves the use of a pool of hidden neurons, each with different initial random starting weights, as in CasCor. Each hidden neuron in the pool is trained using SARPROP, as in the first extension. The network giving the smallest RMS Error value is selected once all hidden neurons complete training. In the case where one or more hidden neurons manage to learn the problem to satisfaction, the hidden neuron with best performance on the test set is chosen. It should be noted that in CasPer the whole network must be trained for each hidden neuron in the pool, so this extension will be computationally expensive. This extension will be term SP_CasPer.

Two minor modifications were also made to the original CasPer algorithm. First, training was started with no hidden neurons. The reason for this change is that many problems can be satisfactorily solved with no hidden neurons. Second, the time period over which hidden neuron training is performed was simplified from 15+L*N to L. This was done to allow the SA term to act over a longer period (a larger L value is used than in the original CasPer). These modifications were used by both S_CasPer and SP_CasPer.

4 COMPARITIVE SIMULATIONS

To test the effectiveness of S_CasPer and SP_CasPer, their performance was compared against that of CasPer and CasCor on a number of benchmark problems. CasPer, S_CasPer, and SP_CasPer share a number of parameters which were set as follows. The following (standard) RPROP values were used: $\eta^+ = 1.2$, $\eta^- = 0.5$, $\Delta_{max} = 50$, $\Delta_{min} = 1x10^{-6}$. A constant value of 0.0001 was added to the derivative of the sigmoid in order to overcome the 'flat spot' problem, and the hyperbolic arctan error function was used [7]. Weights in the initial network were initialised to evenly distributed random values in the range -0.7 to 0.7. All weights associated with newly inserted hidden neurons were initialised in the range -0.1 to 0.1.

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 of the above parameter values were found to be essentially problem independent, and hence are treated as constants. The remaining parameter values *D* (the weight decay value) and *L* (the training length), were set depending on the problem. CasPer halts training when network outputs for all patterns are within 0.2 of the required training outputs, in which case the training set was considered completely learnt. This more restricted value (0.4 being the more traditional value [7]) was chosen since it was found that it improved CasPer's performance on the test set, although it did result in additional training time, and sometimes in larger networks. S_CasPer and SP_CasPer use the same criterion. For judging success on the test set for classification problems the

traditional criterion [7] was used: a pattern was considered correct if all its outputs were within 0.4 of the required outputs.

For S_CasPer and SP_CasPer, the SARPROP parameters used were: $k_1 = 1$, and $k_2 = 0.4$. SP_CasPer used a pool of eight hidden neurons. 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 [2] for the two spirals data set. All other CasCor parameters were kept at the default values.

4.1 TWO SPIRALS DATA SET

The first data set in the comparison was the two spirals problem, consisting of two interlocked spirals, each made up of 97 points, which the network must learn to distinguish. Each training pattern consists of two inputs (the x,y coordinates) and a single output (the spiral classification). This problem was used by Fahlman [2] to demonstrate the effectiveness of the CasCor algorithm on a problem known to be very difficult for traditional Back Propagation to solve. 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 data set (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 $L = 5$ and $D = 5\text{x}10^{-3}$. S_CasPer and SP_CasPer used $L = 100$ and $D = 1\text{x}10^{-2}$. The standard symmetric sigmoid non linearity (-0.5, 0.5) was used. Since the two spirals problem is an artificial data set without the presence of noise, training was continued until the training set was learnt completely. At this point the mean, standard deviation and median for the following characteristics were measured: epochs trained, hidden neurons inserted, number of connection crossings and percentage correct on the test set. Fahlman [2] 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 which greatly improves the algorithm’s efficiency. It should be noted that in CasCor an epoch is defined as single pass of all training patterns through all the candidate neurons, and is not calculated on a per candidate basis, as it is for SP_CasPer. For this reason also, connection crossings give a much better indication of computational cost.

The results for CasPer, S_CasPer, SP_CasPer and CasCor on the two spirals problem are shown in Table 1. S_CasPer was able to produce networks with two less hidden neurons in the median case than CasPer, which equates to 31 less weights being used because of the cascade architecture. S_CasPer also gave improved test set results. One reason for this improvement may be due to the smaller networks created by S_CasPer. In terms of connection crossings S_CasPer and CasPer were approximately equal.

SP_CasPer showed moderately improved performance over that of S_CasPer, with SP_CasPer producing networks with 10 hidden neurons in the median case. SP_CasPer is thus producing networks in the median case which use less than half the number of weights than those of CasCor (88 weights compared to 187). SP_CasPer also produced a slight improvement in the test set results compared to S_CasPer. As expected, SP_CasPer was more computationally expensive, with it performing approximately six times as many connection crossings as S_CasPer.

	Epochs	Hidden Neurons	Conn. Crossings	Test Set %
CasPer				
Average	2437	13.49	9.57×10^7	97.80
Median	2307	13.00	8.56×10^7	98.44
Std. Dev.	627	2.41	5.37×10^7	2.22
S_CasPer				
Average	4392	11.64	1.12×10^8	98.38
Median	4294	11.00	1.03×10^8	98.96
Std. Dev.	755	2.34	4.38×10^7	1.73
SP_CasPer				
Average	27445	10.35	6.07×10^8	98.81
Median	26593	10.00	5.42×10^8	98.96
Std. Dev.	5797	2.29	3.17×10^8	1.38
CasCor				
Average	1686	15.96	2.02×10^7	96.13
Median	1689	16.00	1.99×10^7	96.35
Std. Dev.	209	2.17	4.47×10^6	2.11

Table 1. Results on the Two Spiral data set.

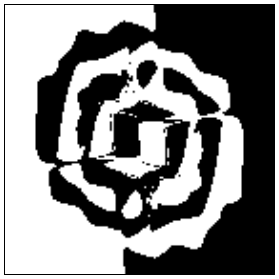

Fig. 2. CasCor.

Fig. 3. CasPer.

Fig. 4. S_CasPer.

For the two spirals data set, CasCor produced 3 networks which gave 100% correct on the test set. CasPer, S_CasPer and SP_CasPer produced 30, 34 and 41 of these networks respectively. Of those networks producing 100% on the test set, Figures 2, 3 and 4 are plots produced from CasCor, CasPer and S_CasPer runs, which illustrate

qualitatively both CasPer and S_CasPer's better generalisation compared to CasCor even in cases where a 100% correct result was achieved on the test set.

4.2 IRIS DATA SET

Fisher's classic Iris data, which classifies irises into three classes, was used for the next comparison. The Iris data set was obtained from the UCI database [8], and consists of 150 patterns, of which 120 were randomly selected as training patterns, leaving 30 as test patterns. Each Iris pattern consists of 4 input and 3 output values. The parameter values used for the CasPer algorithm were $L = 5$ and $D = 1x10^{-5}$, while S_CasPer and SP_CasPer used $L = 100$ and $D = 1x10^{-3}$. The asymmetric sigmoid (0, 1) non linearity was used, since the output values of this data set are in the range 0 to 1. One hundred separate training runs with different initial weight values were performed for the Iris data set, and the results are shown in Table 2.

	Epochs	Hidden Neurons	Conn. Crossings	Test Set %
CasPer				
Average	326	4.01	2.89×10^6	88.87
Median	316	4.00	2.62×10^6	90.00
Std. Dev.	72	1.02	1.07×10^6	3.75
S_CasPer				
Average	1227	3.46	8.73×10^6	88.97
Median	1194	3.00	7.81×10^6	90.00
Std. Dev.	283	1.44	3.99×10^6	3.94
SP_CasPer				
Average	4462	1.82	2.64×10^7	87.43
Median	4567	2.00	2.70×10^7	86.67
Std. Dev.	799	0.54	6.22×10^6	3.09
CasCor				
Average	431	2.60	1.47×10^6	72.22
Median	430	3.00	1.40×10^6	73.33
Std. Dev.	95	0.69	4.74×10^5	7.10

Table 2. Results on the Iris data set.

The performance of each network on the test set after the addition of each new hidden neuron was measured. A plot of the median values obtained is shown in Figure 5. S_CasPer was again able to reduce the average network size produced compared to CasPer. In addition, Figure 5 shows that S_CasPer's initial performance on the test set was significantly better than CasPer, although the two became approximately the same as additional neurons were inserted. SP_CasPer was able to produce further decreases in network size, although a slight drop in generalisation was produced. It seems likely that this is due to overfitting, since in SP_CasPer the hidden neuron selected is the one which gives the largest decrease in error, resulting in networks which may overfit the data set. The maximum number of hidden neurons inserted by SP_CasPer was 3, hence its cutoff in Figure 5.

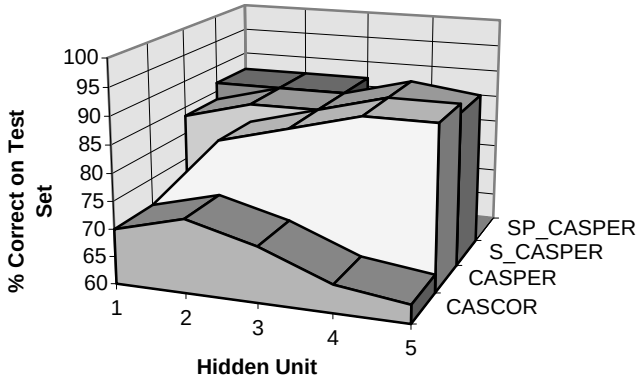

Fig. 5. Performance on the Iris test set after the insertion of each hidden neuron.

The reason for S_CasPer's increased computational cost compared to CasPer in this problem is that the training period for CasPer is initially very small, and only increases as the network grows in size. Since only small networks were produced for this data set, CasPer is less expensive than S_CasPer, as S_CasPer uses a constant value as the training period (100 in this case). Again, SP_CasPer is computationally more expensive than S_CasPer: SP_CasPer performs approximately 3 times as many connection crossings as S_CasPer.

5 DISCUSSION

In both comparisons S_CasPer is shown to produce decreases in network size compared to CasPer. This can attributed in the main to the new hidden neuron being trained by SARPROP, which allows this neuron to perform a better search of the error surface. S_CasPer is generally able to maintain, and sometimes improve the good generalisation ability of CasPer.

Similar improvements are made by the SP_CasPer algorithm. The amount of improvement in hidden neurons inserted by SP_CasPer over S_CasPer, while significant, is not in proportion to the amount of extra computational cost involved in performing the additional search provided by using a pool of hidden neurons. One reason for this may be that the search performed by S_CasPer is very successful at finding a good solution, and this is not much improved even when additional searches are performed. An advantage of SP_CasPer, however, is that it is ideally suited for parallel implementation, with each network in the pool trained on a separate processor. This largely overcomes the additional time required to train a pool of neurons.

The ability to create cascade networks with smaller numbers of hidden neurons is especially relevant to the area of VLSI implementation of these networks. Cascade networks result in deep networks with large fan-in and propagation delays [2]. Smaller networks reduce these difficulties. One problem not addressed by these improvements to CasPer is that the size of the constructed network is difficult to estimate prior to

training. A VLSI implementation, however, will need to set an upper bound on both fan in and network depth. Future work with CasPer is aimed at addressing this problem.

6 CONCLUSION

The S_CasPer extension of the CasPer algorithm, which uses SARPROP as the training method for the newly inserted hidden neurons, results in smaller networks, often with better generalisation. The SP_CasPer extension, which uses of a pool of hidden neurons trained by SARPROP, results in further improvements in network size, although there is an increased computational cost. Additional comparisons between CasPer, S_CasPer and SP_CasPer using regression benchmarks [9] support the conclusions drawn here.

REFERENCES

[1] Treadgold, N.K. and Gedeon, T.D. "A Cascade Network Employing Progressive RPROP", *Int. Work Conf. on Artificial and Natural Neural Networks*, pp. 733-742, 1997.

[2] Fahlman, S.E. and Lebiere, C. "The cascade-correlation learning architecture," *Advances in Neural Information Processing*, vol. 2, D.S. Touretzky, (Ed.) San Mateo, CA: Morgan Kauffman, pp. 524-532, 1990.

[3] J. Hwang, S. You, S. Lay, and I. Jou, "The Cascade-Correlation Learning: A Projection Pursuit Learning Perspective", *IEEE Trans. Neural Networks* **7**(2), pp. 278-289, 1996.

[4] T. Kwok and D. Yeung, "Experimental Analysis of Input Weight Freezing in Constructive Neural Networks", *Proc IEEE Int. Conf. On Neural Networks*, pp. 511-516, 1993.

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

[6] Riedmiller, M. and Braun, H. "A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm," *Proc IEEE Int. Conf. on Neural Networks*, pp. 586-591, 1993.

[7] Fahlman, S.E. "An empirical study of learning speed in back-propagation networks," Technical Report CMU-CS-88-162, Carnegie Mellon University, Pittsburgh, PA, 1988.

[8] Murphy, P.M. and Aha, D.W. "UCI Repository of machine learning databases," [http://www.ics.uci.edu/~mlearn/MLRepository.html], Irvine, CA: University of California, Department of Information and Computer Science, 1994.

[9] Treadgold, N.K. and Gedeon, T.D. "Extending CasPer: A Regression Survey", *Int. Conf. On Neural Information Processing*, to appear, 1997.