Parallel Evolution of Cascade Structures

T.D. Gedeon [1], X. Yao [2], N.K. Treadgold [1] and Y. Liu [2]
School of Computer Science and Engineering [1]
School of Computer Science, University College [2]
The University of New South Wales
Sydney 2052 AUSTRALIA

Abstract

Neural network solutions to practical problems have found many applications. A hindrance to more widespread use is the difficulty in producing good solutions, requiring a great deal of expertise on the part of the neural network designer. A major problem is the design of the network architecture and topology. This paper describes two major strands of research we have undertaken to solve this problem, using a contructive neural network enhanced with simulated annealing, and an evolutionary programming approach. The two techniques produce very similar results, this parallel evolution of essentially the same structures provides a high degree of confidence that these are the optimal structures to be found within the time-space computing bounds we allow.

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, 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.

Growing cascades using Casper

Casper (Treadgold and Gedeon, 1997) 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 (Falhlman and Lebiere, 1990): 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, rather that Cascor's use of Quickprop. 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.

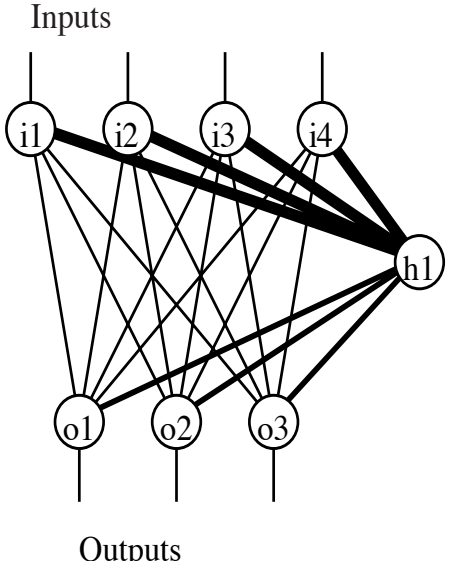

Figure 1. A hidden unit has been added

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 is 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, 1997) 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:

$$dE/dw_{ij} = dE/dw_{ij} - k^* \text{ sign}(w_{ij})^* w_{ij}^2 * 2^{-0.01*HEpoch}$$

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. Note that the version of Casper used for the results below has no user settable parameters. For complete details on Casper, see Treadgold and Gedeon (1998).

Growing cascades using EPnet

EPnet (Yao and Liu, 1997) is based on Fogel's (1991) evolutionary programming, and evolves fully connected cascade structures trained using Quickprop, using a parsimonious scheme with 5 mutations applied in a sequential fall through manner. These are illustrated in Figure 2.

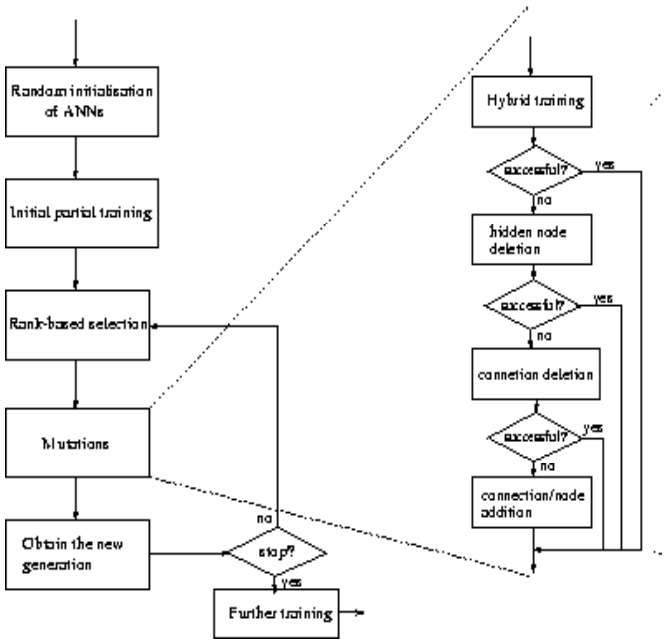

Figure 2. EPnet major steps

The direct encoding scheme is used in Epnet to represent the architectures and connection weights and biases, as two equal size matrices and one vector. One matrix is the binary connectivity matrix, while the second is a real number matrix containing the actual weights. The binary vector represents the presence or abscence of each hidden neuron.

Fitness evaluation is by the use of Prechelt's (1994) error measure on a validation set. This error measure reduces dependence on the size of the validation set and the number of output neurons. The selection mechanism used is rank based, with individuals selected probabilitically and mutated using the five mutations (connection and node addition are separate mutations).

The replacement strategy is conservative, and maintains behavioural links between parent and offspring networks. In hybrid training, two offspring are produced, by further neural training, and by the use of a simulated annealing (SA) algorithm. If a better offspring is not produced at each stage, the next stage is attempted. If an offspring is produced by neural training the parent is replaced. If by the SA algorithm then the parent is only replaced for significant error reduction. Offspring from deletions (connections or neurons) replaces the worst individual only if it is better, while offspring from additions always replaces the worst individual as it has greater computing power and may not have been trained enough. Thus, only when further training will not improve performance are architectural modifications undertaken. For full details of the user parameters set for EPnet, see Yao and Liu (1997).

Comparisons

We will compare the Casper and EPnet techniques in terms of the network topologies produced, along with some standard techniques in terms of the quality of results. We will use two benchmark data sets from the Proben1 repository (Prechelt, 1994), being the diabetes and thyroid data sets.

data set		Casper		EPnet		Cascor
		Number of connections	No. hidden neurons	Number of connections	No. hidden neurons	No. hidden neurons
Cancer	Mean	88.4	**4.9**	41.0	**2.0**	**5.2**
	SD		2.1	14.7	1.1	2.1
	Min	32	1	15	0	3
	Max	144	8	84	5	10
Diabetes	Mean	54.0	**3.0**	52.3	**3.4**	**9.8**
	SD		1.6	16.1	1.3	5.3
	Min	29	1	27	1	0
	Max	134	8	87	6	25
Heart	Mean	75.8	**0.1**	92.6	**4.1**	**1.4**
	SD		0.4	40.8	2.1	0.5
	Min	72	0	34	1	1
	Max	149	2	213	10	2
Thyroid	Mean	189.5	**4.6**	219.6	**5.9**	**25.0**
	SD		2.3	74.4	2.4	8.7
	Min	91	1	128	3	2
	Max	294	8	417	12	44

Table 1: Topological comparison between Casper and EPnet

Both Casper and EPnet are quite similar, with Casper producing networks some 10 to 20% smaller than EPnet. Note the significantly larger networks created by the Cascor method. The very large number of connections required by Cascor have been elided from the table.

data set		Test set results (% incorrect classifications)		
		Casper	EPnet	Cascor
Cancer	Mean	**1.89**	**1.38**	**1.95**
	SD	0.80	0.94	0.38
	Min	0.57	0.00	1.15
	Max	4.02	4.00	2.87

Diabetes	Mean	**23.14**	**22.38**	**24.53**
	SD	1.26	1.40	1.44
	Min	20.31	19.27	22.40
	Max	27.08	25.00	28.65
Heart	Mean	**18.85**	**16.77**	**19.47**
	SD	1.14	2.03	1.28
	Min	18.67	13.24	18.67
	Max	26.67	19.12	24.00
Thyroid	Mean	**1.67**	**2.11**	**3.03**
	SD	0.26	0.22	1.15
	Min	1.28	1.63	2.11
	Max	2.28	2.63	6.56

Table 2: Generalisation comparison between Casper and EPnet

In terms of generalisation, Casper's performance is intermediate between that of EPnet and Cascor. The differences between the number of hidden neurons required suggests that the thyroid data set is more difficult than the diabetes data set. This is somewhat contrdicted by the observation that on diabetes Casper produced sizes 10% less than EPnet while the error was closer to that of Cascor, while on thyroid a 20% smaller size was closer in error to EPnet.

Conclusion

Both Casper and EPnet produced much smaller networks than the Cascor algorithm, which has significant implications in terms of computation time, as the number of connections goes up exponentially with the number of hidden neurons in cascade networks.

The generalisation results for both Casper and EPnet were better than Cascor. EPnet results were the best overall, as can be expected as it is a global optimisation method. There are significant parallels in the natures of the Casper and EPnet techniques, which we will briefly draw out here.

The Casper algorithm does not freeze weights (unlike Cascor), which can be considered analogous to the EPnet's maintenance of behavioural links from parent to child. If behavioural links were not maintained in EPnet, then its networks would need to be retrained anew each time. This would enhance the view that topology is all important. In Casper, if weight freezing ocurred, then the weights appear paramount, yet they are not changeable and only topological changes can fix any embedded errors. Thus EPnet and Casper converge to a middle ground. The use of simulated annealing in both techniques is similar. Finally, the weight decay used by Casper has its analogy in EPnet's mutation ordering favouring the reduction mutations.

Clearly, there are also significant differences in the approaches also. The most significant are the differences in computational time required (evolutionary approaches are expensive), and that Casper has no user settable parameters, while EPnet has a number to be set appropriately for the given task. Our next step will therefore be to tweak Casper parameters for the comparison datasets, and to attempt to automate selection of EPnet parameters. These we expect will reduce the differences between the results and support our contention that the two techniques are converging on general solutions.

References

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

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.
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.
Fogel, D.B. *System Identification Through Simulated Annealing*, Needham Heights, 1991.
Hwang, J., You, S., Lay, S. and I. Jou, "The Cascade-Correlation Learning: A Projection Pursuit Learning Perspective," *IEEE Trans. Neural Networks* **7**(2), pp. 278-289, 1996.
Kwok, T. and Yeung, D. "Experimental Analysis of Input Weight Freezing in Constructive Neural Networks," *Proc IEEE Int. Conf. On Neural Networks*, pp. 511-516, 1993.
Prechelt, L. "Proben1 - A Set of Neural Network Benchmark Problems," [ftp.ira.uka.de/pub/neuron/proben1.tar.gz], University of Karlsruhe, Germany, 1994.
Riedmiller, M. "Rprop - Description and Implementation Details," Technical Report, University of Karlsruhe, 1994.
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.
Treadgold, N.K. and Gedeon, T.D. "A Cascade Network Algorithm Employing Progressinve RPROP," in Mira, J, Moreno-Díaz, R and Cabestany, J, (eds.), *Biological and Artificial Computation: From Neuroscience to Technology*, pp. 733-742, Springer Verlag, Lecture Notes in Computer Science, vol. 1240, 1997.
Treadgold, N.K. and Gedeon, T.D. "Adaptive Regularisation in a Constructive Cascade Network," *Proceedings International Conference on Neural Information Processing (ICONIP'98)*, 6 pages, Japan, 1998.
Yao, X. and Liu, Y. "A New Evolutionary System for Evolving Artificial Neural Networks," *IEEE Transactions on Neural Networks*, vol. 8, no. 3, pp. 694-713, 1997.