Constructing Higher Order Neurons of Increasing Complexity in Cascade Networks

N.K. Treadgold and T.D. Gedeon
Department of Information Engineering
School of Computer Science & Engineering, The University of New South Wales
{nickt | tom}@cse.unsw.edu.au

Abstract. A problem faced by many constructive neural networks using a cascade architecture is the large network depth. This results in large fan-in and propagation delays, problems especially relevant for VLSI implementation of these networks. This work explores the effect of limiting the depth of the cascades created by CasPer, a constructive cascade algorithm. Instead of a single cascade of hidden neurons, a series of cascade towers are built. Each cascade tower can be viewed as a single Higher Order Neuron (HON). The optimal complexity of the HON required for a given problem is difficult to estimate, and is a form of the bias-variance dilemma. This problem is overcome via the construction of HONs with increasing complexity. It is shown that by constructing HONs in this manner the chance of overfitting is reduced, especially with noisy data.

1 Introduction

Constructive cascade algorithms such as CasPer [1,2] and Cascade Correlation (CasCor) [3] are poorly suited to VLSI implementation because the cascade architecture has large network depth. This results in large propagation delays and high fan-in. The constructed networks also have irregular connections. In addition, the growth of network weights is exponential as more neurons are added. Previous work has looked at overcoming these problems through the construction of a series of cascade towers of fixed depth [4]. Each tower can be viewed as a Higher Order Neuron (HON), the complexity of which is determined by the number of neurons in the tower. A difficulty which is not addressed, however, is the selection the optimal HON complexity for a given problem. More complex HONs are able to fit more complex functions, however they are also more susceptible to overfitting. The optimal choice of HON complexity is a form of the bias-variance dilemma.

This work looks at overcoming this problem by constructing a series of HONs, the complexity of which are increased as training continues. The algorithm incorporating these features will be termed HON_CasPer. This algorithm overcomes the need for prior setting of the HON complexity, and reduces the chance of overfitting. By limiting the maximum HON complexity, this architecture is also made more suitable for VLSI implementation due to limited network depth and a regular connection strategy. In addition, the weight growth per additional hidden neuron becomes essentially linear. A plot of the weight growth per neuron added is shown in Figure 1 for both the traditional cascade architecture and the HON architecture. This

figure illustrates the slower growth in network weights for the HON network, especially as the networks grow in size.

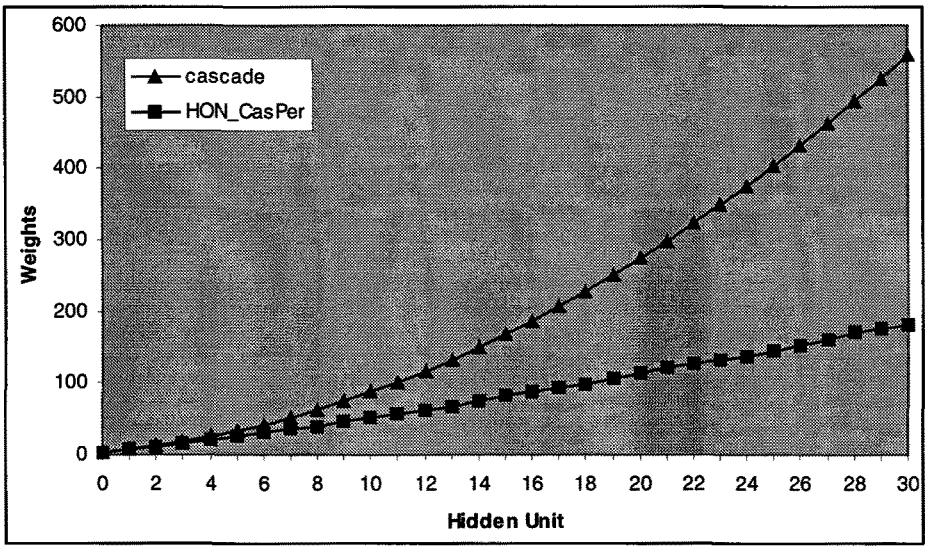

Fig. 1. Weight growth per hidden unit

2 The CasPer Algorithm

CasPer uses a modified version of the RPROP algorithm [5] 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 all inputs connected directly to the outputs, and successively inserts hidden neurons to form a cascade architecture. 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 that depend on the position of the weight in the network. The network is divided into three separate groups, each with its own initial learning rate: *L1*, *L2* and *L3*. 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.

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. In addition, the L1 weights are trained by a variation of RPROP termed SARPROP [6]. The SARPROP algorithm is based on RPROP, but uses a noise factor to enhance the ability of the network to escape from local minima. The amount of noise added falls as training continues via a Simulated Annealing (SA) term.

CasPer also makes use of weight decay as a means to improve network generalization. After some experimentation it was found that the addition of a SA term applied to the weight decay, as used in the SARPROP algorithm, often improved convergence and generalization. Each time a new hidden neuron is inserted, the weight decay begins with a large magnitude (set by a parameter *D)*, which is then reduced by the SA term. 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 parameter *L*.

3 HON Modifications to CasPer

In order to create a series of HONs of increasing complexity, CasPer is modified to construct series of cascade towers. The size of each cascade tower is increased incrementally as training continues, thus producing ever more complex HONs. An example of this architecture is shown in Figure 2. A limit is fixed on the maximum size of HON constructed, after which all further HONs remain at this complexity. The reason for this is that it stops the growth of arbitrarily complex HONs. This enables easier VLSI implementation since the maximum network depth is fixed. A reasonable limit would be of size eight, although this limit is not reached in any of the simulations performed and is only of relevance for VLSI implementation.

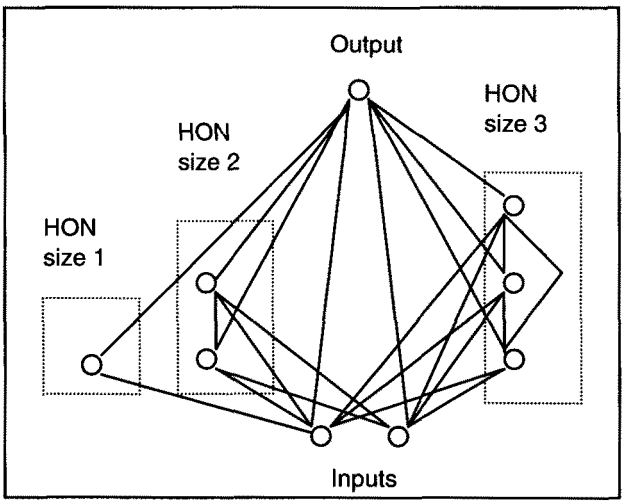

Fig. 2. The CasPer architecture using HONs of increasing complexity

The only modification to the standard CasPer training algorithm is the addition of a 'backfitting' training stage, which is done after the completion of each HON. It was found that the addition of backfitting improved the convergence of the algorithm. The backfitting proceeds by training each HON in turn, starting with the first HON constructed, up to the most recent. The backfitting is achieved using RPROP with the initial learning parameters set as follows. All weights connected to the HON undergoing backfitting (including incoming, outgoing and internal weights) are assigned the initial learning rate T1. All other weights have initial learning rates T2. The values of these constants are set such that T1>>T2. This was done to maximize the ability of the tower undergoing backfitting to adapt to the remaining error, as in the original CasPer training methodology.

4 Comparative Simulations

To investigate the performance of the HON_CasPer algorithm, one classification and two regression benchmark problems were selected. Comparisons were made between the original CasPer and HON_CasPer algorithms. The standard CasPer constant settings were used [1,2] for both CasPer and HON_CasPer. The constants T1 and T2 were set to 0.2 and 0.001 respectively. The parameter values for L and D were set to give the best performance for each algorithm. It was found that in general both algorithms obtained their best performance using similar values.

The first comparison was performed on the two spirals data set which consists of two interlocked spirals. The standard test set for the two spirals data set was used to measure the resulting generalization ability of the networks. This test set consists of two spirals slightly rotated relative to the original spirals. Fifty independent training runs were performed. The parameter values used for the CasPer and HON_CasPer algorithms were L = 100 and D = 0.01. Training was continued until the training set was learnt completely or a maximum of 50 hidden neurons were installed.

At this point the mean, standard deviation and median for the following characteristics were measured: number of connection crossings, hidden neurons inserted, percentage correct on the test set, and the number of network weights. Fahlman [3] 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 measure of computational cost is used instead of epochs since the network sizes are continually changing. These results are shown in Table 1.

In all trials the training set was learnt before the maximum 50 hidden unit limit. HON_CasPer is able to generate slightly better generalization results than CasPer. HON_CasPer, however, can be seen to install many more hidden neurons, although the actual difference in terms of number of weights used in the networks is not as significant. These results are not surprising when the complexity of the problem being solved is considered: the two spirals problem is highly non-linear and networks with more complex HONs are more easily able to solve it. HON_CasPer is initially limited to more basic HONs, and it is only as training proceeds that it constructs HONs of the required complexity to solve the problem. Thus the two spirals problem

can be seen as a worst case scenario for the HON_CasPer algorithm in terms of network size, and correspondingly, convergence speed.

Table 1. Two spirals results

	CONN. CROSS. (10^8)	UNITS	TEST%	WEIGHTS
CasPer				
Average	1.12	11.64	98.38	
Median	1.03	11.00	98.96	102
Std. Dev.	0.44	2.34	1.73	
HON_CasPer				
Average	4.72	25.52	99.22	
Median	4.08	26	99.48	155
Std. Dev.	2.33	5.55	1.08	

Two regression functions were chosen to compare Casper and HON_CasPer. The functions are described in detail in [7], and are shown below:

- Complex additive function:

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

- Complex interactive function:

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

The set up of training and test data follows the method of [7]. For each function two sets of training data were created, one noiseless and one noisy, using 225 random values. The noisy data was created by adding independent and identically distributed Gaussian noise, with zero mean and unit variance, giving an approximate signal to noise ratio of 4 [7]. For each function an independent test set of size 2500 was generated on a regularly spaced grid $[0,1]^2$. The fraction of variance unexplained (FVU) [7], which is proportional to the total sum of squares error, was the measure chosen to compare the performances on the test set. For each regression function 50 runs were performed using different random starting weights. 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 and the median values are plotted in Figures 3 and 4.

CasPer and HON_CasPer show similar results for the noise free data sets. The noisy data sets show HON_CasPer producing better results, and also being less susceptible to overfitting (indicated by a rising FVU). In addition to HON_CasPer producing better generalization results, the actual size of the networks constructed in

terms of weights is significantly smaller due to CasPer's exponential growth of weights. For example, after the insertion of 30 neurons, HON_CasPer uses only 180 weights compared to the 558 weights of CasPer.

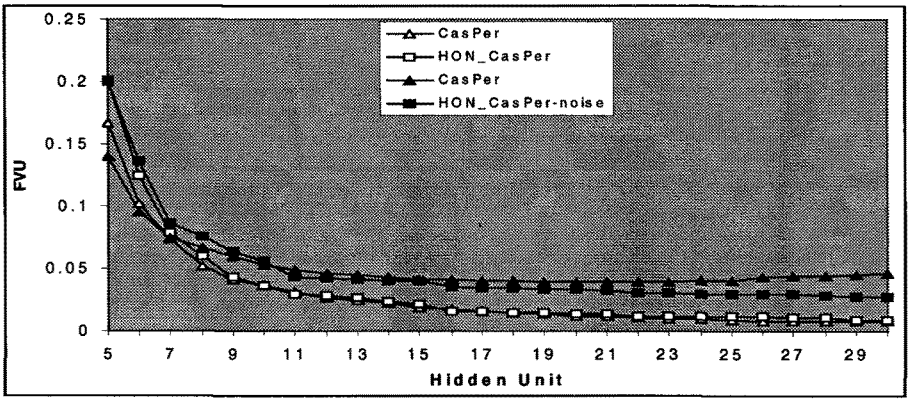

Fig. 3. Complex additive function results

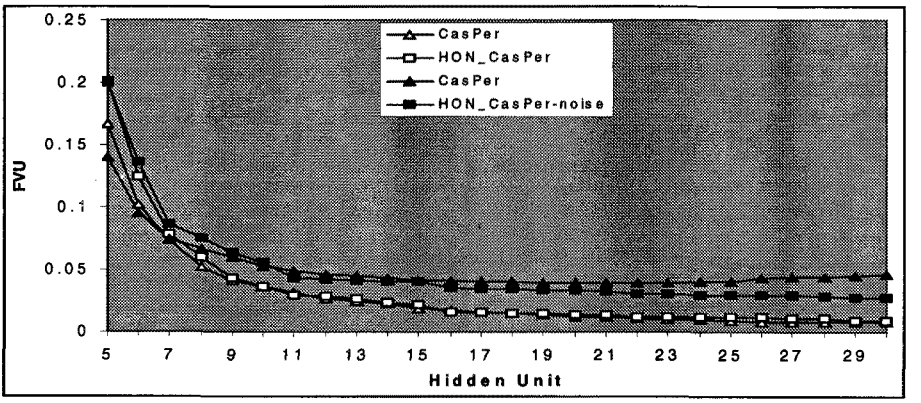

Fig. 4. Complex interactive function results

5 Discussion and Conclusion

One of the main advantages of the introduction of the HON_CasPer architecture is in terms of VLSI implementation. The HON_CasPer architecture allows a maximum network depth to be set, while CasPer has a potentially unlimited depth. This reduction in network depth is demonstrated in the simulation results: for the two spirals problem HON_CasPer produced a median depth of 6, compared to CasPer's 11. In the regression benchmarks, after the installation of 30 neurons, HON_CasPer's depth was 7 compared to 30 for CasPer. The connections in the HON_CasPer

architecture are also much more regular than the CasPer architecture: the HON_CasPer network being made up of cascade towers of a fixed maximum depth.

One problem that is not addressed by the HON_CasPer architecture is the high degree of fan-in, which can be a problem for VLSI implementation. The maximum fan-in using the HON_CasPer architecture is exactly the same as in the original CasPer architecture for a given number of hidden units installed. The fan-in can be greatly reduced in the HON_CasPer architecture, however, by introducing a new linear summing neuron in each tower [4]. This neuron takes as its input the weighted sum of all tower outputs (which were originally connected to the output neurons) and its output is fed directly to the output neurons. This reduces the maximum fan-in from I+N to max(I+HS, I+HN), where I is the number of network inputs (including bias), N is the number of hidden neurons, HS is the maximum HON size and HN is the number of HONs. For example, the maximum fan-in for HON_CasPer after the installation of 30 neurons in the regression problems is 11 compared to CasPer's 33.

In terms of network performance, HON_CasPer is able to maintain and in some cases better network generalization, especially in the presence of noise. This can be attributed to HON_CasPer reducing the chance of overfitting the data by limiting the size of the HON complexity. CasPer, however, constructs a single complex HON which has a greater potential for overfitting.

References

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

2. Treadgold, N.K. and Gedeon, T.D. "Extending CasPer: A Regression Survey," *Int. Conf. On Neural Information Processing,* 1997, pp. 310-313.

3. 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, 1990, pp. 524-532.

4. Treadgold, N.K. and Gedeon, T.D. "Exploring Architecture Variations in Constructive Cascade Networks," Int. *Joint Conference on Neural Networks,* 1998, to appear.

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

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

7. Hwang, J., Lay, S., Maechler, R. and Martin, D. "Regression Modeling in Back-Propagation and Projection Pursuit Learning," *IEEE Trans. Neural Networks* **5**(3), 1994, pp. 342-353.