Extracting Meaning from Cascade Networks

T.D. Gedeon and N.K. Treadgold
Department of Information Engineering
School of Computer Science & Engineering
The University of New South Wales
Sydney NSW 2052 AUSTRALIA
E-mail: tom@cse.unsw.edu.au
Fax: +61 2 9385 5995

ABSTRACT

Cascade networks have advantages over the more familiar layered feedforward neural network architectures in terms of their ability to solve certain problems, and in their automation of the task of specifying the size and topology of network to use.

Cascade networks still share the problem of lack of explanatory mechanism, and remain 'black boxes' sometimes mistrusted by end users. The more complex topologies of cascade networks complicates explanation or rule extraction, hence little previous work has been done. We extend our technique based on clusters of characteristic input patterns using the advantages of an improved cascade network.

1. INTRODUCTION

Neural networks which are grown to the appropriate size to match the requirements of particular problems and data sets have the advantage that the best number of processing elements in a layer do not need to be known in advance or discovered by trial and error but is generated during the problem solution.

Dynamic node creation [1] uses standard back-propagation on small networks and adds new units to the hidden layer when the drop in the error compared to the error when the last node was added falls below some user defined trigger value. New units are fully connected to previous and subsequent layers, with small random weights. The results were generally within 3 units of known minimal solutions.

Statically sized nets at the minimal size at startup often take inordinate amounts of time to train as is well known. Starting with a lower dimensionality network and then increasing its size gradually, a network can spend less time training at maximum size than the equivalently sized back-propagation model. Certainly in the reverse situation where the dimensionality is constricted, there is some significant smoothing and generalisation of the error surface into the lower dimension [2].

The problem of deciding network size is likely to remain difficult in that if the number of hidden units required for a minimal solution could readily be calculated, then this would imply that we could decide what internal representations will be required [3]. We would then be able to solve the problem by some method directly using these internal representations, and would not need the training-by-example capabilities of neural networks. That is, the use of neural networks in general is likely to remain most important in areas in which the then current state of knowledge there remain problems which cannot be solved directly.

Networks with a cascade structure based on the original cascade correlation network [4] have the added advantage that the number of layers of processing elements is also adaptive to the problem being solved.

A disadvantage of cascade correlation is that ill chosen representations are also frozen early and have to be corrected by later layers. This problem is solved by the CasPer algorithm [5] which produces smaller networks which generalise better and do not perpetuate early network artefacts.

In previous work [6] we extracted explanations in the context of characteristic clusters of input patterns based on (standard feedforward) network output classification. Acceptable results were produced even with rude clustering based on averaging pattern values. CasPer cascade networks provide us with the opportunity to produce clusters of input patterns which are hierarchically sorted based on the cascaded structure. Explanations are produced in the context of these patterns.

2. CASCADE ARCHITECTURE

The cascade correlation [4] architecture starts from a simple structure which consists only of the inputs linked directly to the output units.

The output units are trained as far as possible, which is

0-7803-4053-1/97/$10.00 © 1997 IEEE

detected by lack of significant error reduction over some number of epochs.

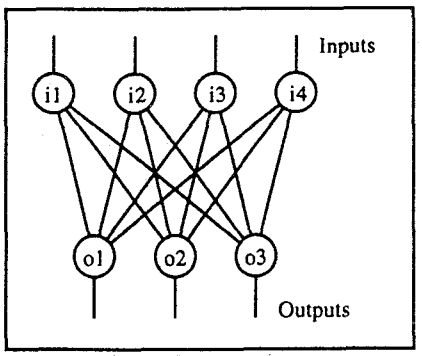


Figure 1. Initial cascade net.

Another pass is made over the training set to measure the error, and terminate the training if the residual error is sufficiently low. Alternatively, a new unit is added to the network, selected from a pool of candidate units.

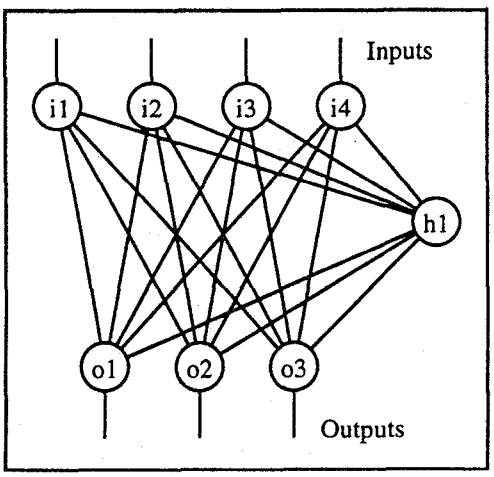


Figure 2. Cascade net – 1 hidden unit.

The pool of candidate units are trained for a number of epochs with inputs of the networks external inputs as well as the results from any existing non-output units. The new unit selected is the one whose output is best correlated with the residual error. New units are connected as they are trained to all external inputs and to the outputs of all other non-output units. All training is done using the QuickProp algorithm [7].

The advantages of the cascade correlation (CasCor) algorithm are that the majority of the network is frozen, and there is on one layer of weights being trained at any one time. This has the advantage of reducing the herd effect in that only candidate units are free to respond to error information. Candidate units in the pool can also attempt to learn the error signal without the competition/interaction with other units, as there are no lateral connections within the pool, nor secondary 'connections' via effects on the error term as the pool is observational only, units within it cannot reduce the error measure.

While CasCor has been shown to be very successful [4], two drawbacks have been observed. These are over-large networks, and generalisation problems. The first of these is due to the weight freezing [8], as this 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. The second drawback of poor generalisation on regression and some classification problems [9] is explained 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. In addition, the correlation mechanism also forces the network to closely match the training set, which may cause overfitting in data sets which have noise present.

We believe that weight freezing may also be a factor in the poor generalisation abilities of CasCor trained networks, since any early hidden units which act as poor feature detectors are frozen. In this case, again the network requires further units to fix the errors introduced by the earlier units, making it difficult to produce a smooth output function.

3. CASPER ALGORITHM

In this paper we will make use of a new cascade network algorithm employing Progressive RPROP (CasPer) which we proposed previously [10], hence it is briefly described here.

CasPer uses a modified version of the RPROP algorithm [11-12] 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 update value for each weight is modified so that if the gradient direction has remained the same, then the update value is increased, while if the gradient has changed it is decreased. The actual value of the gradient is not used, only the direction or sign of the gradient.

The RPROP algorithm has a number of advantages. It is fast to converge compared to the Back Propagation (BP) algorithm [13] and a number of other BP variants, and its performance is relatively invariant to initial parameter selection [12]. It is also only slightly more computationally complex than BP.

The CasPer algorithm constructs cascade networks in a similar manner to CasCor: CasPer starts with a single

hidden unit and successively inserts single hidden units. RPROP is used to train the whole network each time a hidden unit is added. The use of RPROP is modified, however, such that when a new unit 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. The first group is made up of all weights connecting to the new unit from previous hidden and input units. The second group consists of all weights connecting the output of the new unit to the output units. 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 units.

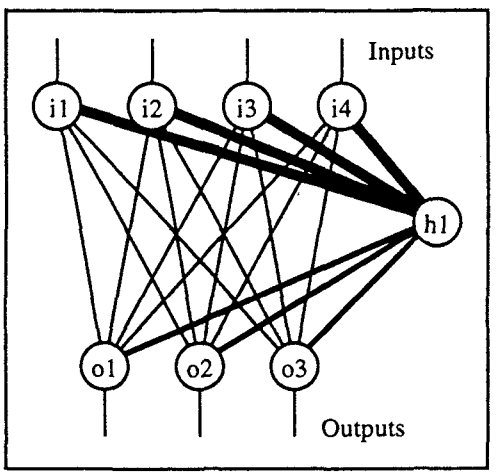


Figure 3. CasPer weights: L1 » L2 > L3

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 unit to learn the remaining network error. Similarly, having L2 larger than L3 allows the new unit 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 unit. Thus CasPer retains the effective benefits of the weight freezing and correlation techniques of CasCor, while removing both the saturation problem caused by the correlation measure, and the permanency of any poorly performing units caused by weight freezing. The removal of these two problems results in better network generalisation.

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 [10] 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 square of the weight magnitude, which results in larger weights decaying more rapidly.

In CasPer a new unit is inserted once the RMS error falls by less than 1% of its previous value. The time period over which this measure is taken is affected by the size of the network – CasPer increases the period over which the network is trained as the network grows in size.

4. EXPLANATIONS - CHAR. PAT.

Many explanation and rule extraction techniques have been attempted for neural networks, with varied success. Many of these techniques are based on static properties of networks. We have shown that this approach is less useful than behavioural measures when examining the functionality of individual hidden units in a network [14]. We have also recently shown that functional measures are better than static measures in the related domain of data mining of inputs [15]. We can speculate that there may be many weight matrix configurations giving rise to similar unit / network behaviour over the range of input patterns likely to be encountered.

Our technique [6] is based on the use of causal index connections between inputs and outputs. This notion was introduced [16-17] using an assumption of a constant value for part of the derived formula of the rate of change of an output unit y_k with respect to an input unit x_i using the chain rule of differentiation. This assumption left terms based only on the static weight matrix. While this approach seemed to work for some data sets, we found that the assumption did not hold for any data sets we examined. This is in accord with our subsequent investigations of static versus dynamic properties of neural networks.

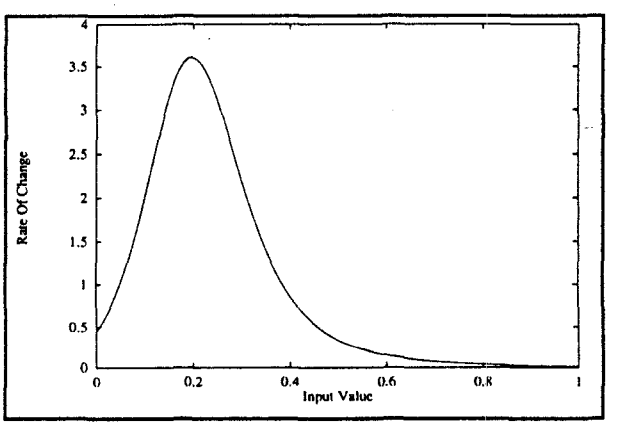


Figure 4. The product $f'(U_{k2}).f'(U_{j1})$ is not constant.

We use special input patterns called *characteristic patterns* to reduce the search space from needing to examine network behaviour on all input patterns.

As we are interested in explaining how a network comes to a conclusion, we select characteristic patterns based on the network behaviour, rather than the classifications in the labelled set of patterns. Thus, the *characteristic on* (*Con*) set of patterns is those for which a particular output unit (category) is turned on. For very simple networks, we can similarly form a *characteristic off* (*Coff*) set of patterns which turn the output of that unit off. For most more complex networks, we simply use the *Con* sets of other outputs as a number of *Coff* sets.

We produce concise explanations as follows:

1. Liken the input pattern to the characteristic input patterns, and present the most similar to the user.
2. In addition, present inputs considered 'important' for the current network output, and their values in the characteristic pattern.
3. Produce a set of rules to confirm accuracy.
4. Give the network's next most likely output.

The first step of the explanation can be compared to some forms of explanations used by human experts to explain their conclusions. As an example consider a doctor explaining why he has come to a particular diagnosis. A typical explanation may include statements such as "You have all the classic symptoms of X." Presenting the characteristic input pattern is similar to this kind of behaviour.

As there may be more than one characteristic input pattern produced from a network's training set (one for each distinctive output) the input pattern is compared to all these patterns and the most similar characteristic pattern is presented.

Once the correct characteristic input pattern has been found it is a simple operation to present the inputs important in the current input pattern. The inputs appearing in the graph of this pattern are presented to the user, with their characteristic values. This is done even in cases in which the pattern is being used as a characteristic *OFF* pattern for another output. This procedure can be likened to the manner a doctor explains a diagnosis he has made. The patient has most of the standard symptoms of a certain disease *X*, however, one or more different symptoms leads the doctor to the conclusion that the diagnosis is disease *Y*.

In some cases (such as that described above), patterns are similar to that of one characteristic pattern, but result in a different output. In these cases, rules are presented to provide an invaluable insight into how the network is making its decisions. In other cases the rules produced can offer some help in understanding the network's actions.

To select the next most likely output, the simple comparison used in the choice of characteristic inputs is again used. In this case however, the next most likely output is that whose characteristic *ON* input pattern is most similar to the current input pattern, other than the characteristic input pattern for the network's current output. This method produces the output (other than the current output) that will occur by making the smallest possible change to the input pattern.

5. EXPLANATIONS – CASCADE NET

The cascade networks produced by CasPer do not freeze weights, thus even the earlier weights in these networks contribute to the final network in a coherent manner. We can disentangle the behavioural effects of the weights from each part of the network in their contribution to the output to determine a hierarchical classification scheme starting with the weights corresponding to the initial zero hidden units case, followed by one hidden unit and so on up to the maximum number of cases.

In this paper, however, we have opted for the simpler solution which is to checkpoint the weight matrix values at each stage and perform our analysis using these networks. Note that both methods are applicable to cascade networks produced by other training algorithms, however, this simpler solution reduces the smoothing benefit we get from our use of CasPer and hence this the conservative choice in terms of comparison to other techniques.

Explanations are produced in the context of these clusters of similar patterns.

The data set we have used is Fisher's classic Iris data set in which irises are classified into three classes. It 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.

6. RESULTS

The Iris results (Figure 12 and Table 2) support the notion that CasCor does not perform well on classification tasks in the presence of noise, while CasPer is shown to maintain a good level of generalisation with such data sets. An interesting point to note in Figure 12 is that CasCor produced no networks larger than 5 hidden units, while CasPer continued to add up to 6 hidden units. This is reflected in Table 1.

Table 1. CasPer versus CasCor on Iris data set.

	Epoch	Hidden Units	Conn. Cross.	Test Set %
CasPer				
Median	316	4.00	2.62 x 10^6	90.0
Std. dev.	72	1.02	1.07 x 10^6	3.8
CasCor				
Median	430	3.00	1.40 x 10^6	73.3
Std. dev.	95	0.69	4.7 x 10^5	7.1

CasPer inserts on average of 4.01 hidden units, as compared to CasCor's 2.6. The result of CasCor's quick learning of the training set is that it overfits the training data, resulting in poor and decreasing performance on the test set. It should be noted that with this type of noisy classification problem it is results on the test set which will define when training should be halted, and not at what point the training set is actually learnt. According to this criterion CasPer produces networks which, if training were halted after the insertion of any hidden unit, both perform well on the test set, and in general would outperform the corresponding CasCor networks as demonstrated in Figure 5.

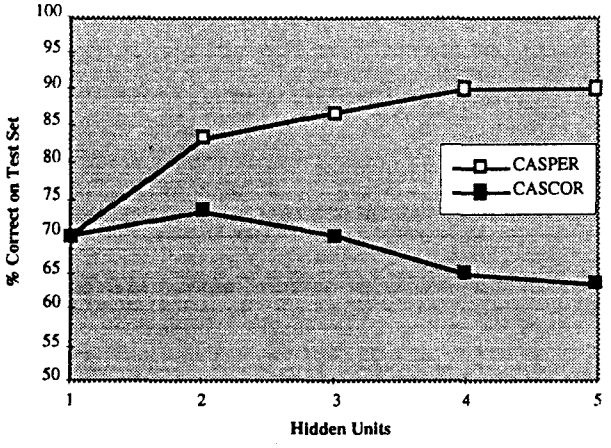


Figure 5. CasPer learns Iris, CasCor overfits.

We have chosen a single CasPer net which terminated training on 3 hidden units (as most comparable to CasCor) for extracting explanations. The explanation rules only are shown in Table 2.

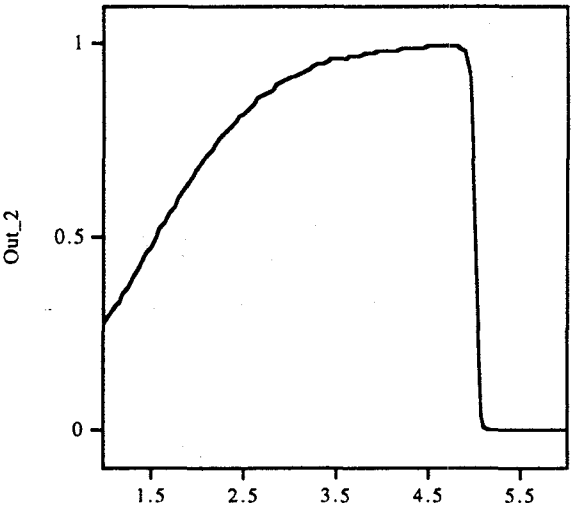
Note that there are three sets of rules in Table 2 (above, right). For the sake of brevity, the complex rules in the *Units = 1* case have been elided except for the rules for the second output unit.

The table shows that the network initially forms some rough characterisation of the inputs, but there are exceptions. That is, we must remember that these rules are expressed in terms of deviations from similarity of the characteristic patterns for each output unit. To be in the class represented by *Out_1*, a pattern must be most similar to that characteristic pattern, and obey the rule.

Table 2. Explanation rules CasPer 1 to 2 units.

Units = 0	
Out_1	In_3 < 3.36
Out_2	In_2 < 3.03 & In_3 > 2.49 & In_4 < 1.58
Out_3	In_3 > 1.42 & In_4 > 1.75
Units = 1 (elided rules for Out_1, Out_3)	
Out_2	In_1 > 4.97 & In_2 < 3.91 & 1.56 < In_3 < 5.00 & In_4 < 2.28
Units = 2	
Out_2	In_1 < 5.88 & In_2 < 3.17 & 4.82 < In_3 & 1.82 < In_4

We can see that in the single example shown in Table 2 for the *Units = 1* case, the rule is more complicated. In fact, all of the rules are more complicated than for the initial case. Of course, the network performance has most likely increased at the same time as shown in the aggregate results depicted in Figure 5. The single example shown also shows the only occurrence of a central subrange, for *In_3*, in this rule set.



By the time the network converges to its best solution, in this case with only 3 hidden units, the rules are much

simpler. That is, the rules for both classes represented by *Out_1*, and *Out_3* are simply for patterns being most similar to the respective characteristic patterns. All of the exception cases are concentrated on one outputs class. The patterns which are most similar to *Out_2* must obey the rules listed, or are not of that represented class and hence are of the class of the next most similar characteristic pattern.

7. CONCLUSION

We have described the extension of our technique of providing explanations for neural network conclusions to networks with cascade architectures. Cascade networks, particularly those produced using our CasPer algorithm, provide us with the opportunity to produce clusters of input patterns which are hierarchically sorted based on the disentangling or checkpointing the cascaded structure. Rules are produced in the context of these clusters of similar patterns.

8. REFERENCES

[1] Ash, T, "Dynamic node creation in backpropagation networks," ICS Report 8901, University of California, San Diego, 1989.

[2] Kruschke,JK, "Improving generalization in back-propagation networks with distributed bottlenecks," *Proceedings International Joint Conference on Neural Networks*, vol. 1, pp. 443-447, 1989.

[3] Harris, D and Gedeon, TD "Adaptive insertion of units in feed-forward neural networks," *Proceedings 4th International Conference on Neural Networks and their Applications*, 9 pages, Nîmes, 1991.

[4] Fahlman, SE, Lebiere, C, "The cascade-correlation learning architecture," CMU-CS-90-100, Carnegie Mellon University, 1990.

[5] Treadgold, NK and Gedeon, TD "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.

[6] Gedeon, TD and Turner, H "Explaining student grades predicted by a neural network," *Proceedings International Joint Conference on Neural Networks*, pp. 609-612, Nagoya, 1993.

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

[8] Kwok, T and Yeung, D "Experimental Analysis of Input Weight Freezing in Constructive Neural Networks," Proceedings IEEE Internetional Conference on Neural Networks, pp. 511-516, 1993.

[9] Hwang, J, You, S, Lay, S and Jou, I "The Cascade-Correlation Learning: A Projection Pursuit Learning Perspective," *IEEE Transactions on Neural Networks*, vol. 7, no. 2, pp. 278-289, 1996.

[10] Treadgold, NK and Gedeon, TD "A Simulated Annealing Enhancement to Resilient Backpropagation," *Proceedings International Panel Conference on Soft and Intelligent Computing*, Budapest, pp. 293-298, 1996.

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

[12] Riedmiller, M "RPROP - Description and Implementation Details," Technical Report, University of Karlsruhe, 1994.

[13] Rumelhart, DE, Hinton GE and Williams RJ "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1, MIT Press, pp. 318-362, 1986.

[14] Gedeon, TD "Indicators of Hidden Neuron Functionality: the Weight Matrix versus Neuron Behaviour," *Australasian Journal of Intelligent Information Processing Systems*, vol. 3, no. 2, pp. 1-9, 1996.

[15] Gedeon, TD "Data Mining of Inputs: Analysing Magnitude and Functional Measures," *International Journal of Neural Systems*, 11 pages, 1997.

[16] Hora, N, Enbutsu, I and Baba,K "Fuzzy rule extraction from a multilayer neural net," *Proceedings IEEE*, vol. 2, pp. 461-465, 1991.

[17] Yoda, M, Baba, K and Enbutsu, I "Explicit representation of knowledge aquired from plant historical data using neural networks," *International Joint Conference on Neural Networks*, San Diego, vol. 3, pp. 155-160, 1991.