Analyser Neural Networks: An empirical study in revealing regularities of complex systems

A.F. Nejad and T.D. Gedeon

School of Computer Science and Engineering
The University of New South Wales
Sydney 2052 AUSTRALIA
{tom | akbar}@cse.unsw.edu.au

Abstract

The causal index method is one of the most applied techniques in extracting meaning from neural networks. We have derived a measure of the significance coefficients for input variables using the weight distribution in the first layer of back-propagation trained multi-layer perceptrons, called *hidden indexes* which we compare to causal indexes. We have also studied finding a degree of importance for each input value depending on its coefficient in a system equation represented as in a multivariate linear regression model, by passing the weight matrix of a trained network to another network. This is an *analyser* neural network.

1 Introduction

One of the most successful attribute based learning methods are artificial neural networks (ANNs). In spite of their advantages, if we accept data transparency as a comparison component, they do not provide any rule or explanation for the decision for an input pattern. That is one of the main reasons for which ANNs are criticised by researchers in other fields. Thus, extracting rules and explanations is an important area of research in neural networks.

Previous work on extracting meaning from neural networks can be group

categorised in three groups: (i) generating a set of rules; (ii) interpreting a trained neural network by visualisation techniques; and (iii) calculating a significance factor for input variables.

The first group of researchers have tried to generate or couple ANNs with a set of consistent and generalised rules [1-6]. Their aim has been to produce a sequence of rules as produced by algorithms like CART [7], or M5 [8] which can be consulted by users to explain why a decision has been made. The second group have worked on analysing weight matrices and activation of hidden units by visualising their changes during training [9-11], while the third group attempt to assign some level of significance to each input pattern variable by calculating the derivative of an output neuron with respect to an input variable (causal index) [12-13], or measuring the change in an output unit by a small change in an input variable (sensitivity analysis) [14]. They assign some coefficients to each independent input variable to the ANN which may be compared with statistical methods like standardised canonical discriminant function coefficients.

The causal index has some special importance in that these results may also be used in other methods [eg. 6, 12]. However, these indexes are sensitive to the hidden layer size and to the error criterion used to terminate training.

Figures 1 and 2 show the causal index (CI) dependency on the structure and parameters in an ANN trained to predict final students marks by the error back propagation algorithm.

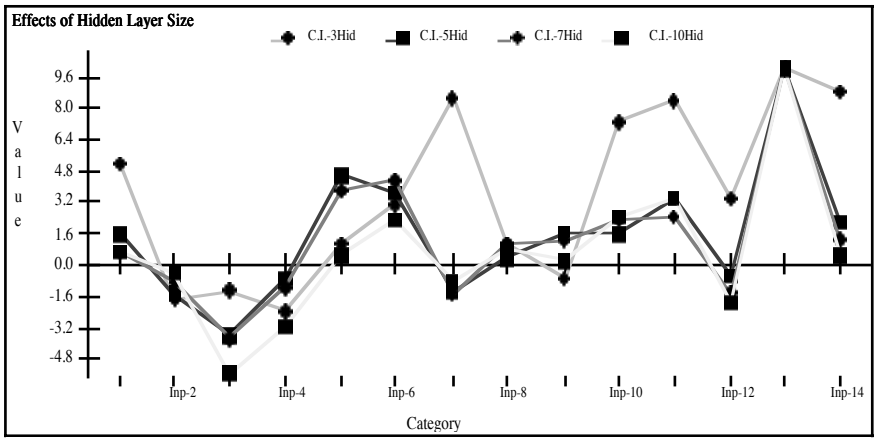

Figure 1. Difference among Causal Indexes of trained neural networks to predict student final marks with 3, 5, 7, and 10 hidden units.

This paper uses a new method to find significant coefficients. Comparing this method with causal index and sensitivity analysis (SA) not only measures its correctness but also may reveal some regularities of the behaviour of complex systems. We also investigate using an ANN in assigning significance indexes to

each input of any other ANN by using its raw weight matrix as training input.

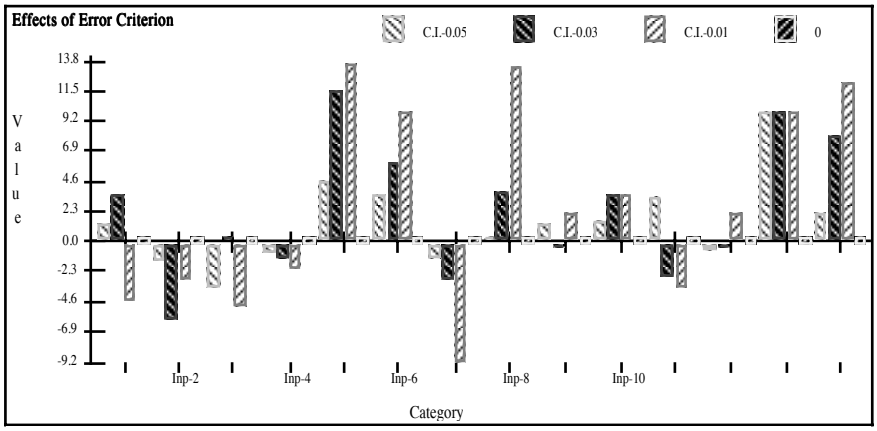

Figure 2. Relative magnitude of causal indexes may indicate the equivalent point in trade-off between overfitting and generalisation.

2 Training Analyser Neural Networks

For extracting meaning from artificial neural networks, we try to find some way to analyse the network by assigning a multi-variable equation to input patterns. Much of the previous work in this area was done on extracting logical rules from neural networks. We will find a degree of importance for each input value depending on the coefficient which will be shown in the equation.

Suppose we have found a multivariate regression equation like $y = a1\ x1 + a2\ x2 + a3\ x3$ for our network. Using this equation we will be able to describe the behaviour of the output value and the degree of positive or negative dependency on each of the inputs.

Limiting the coefficients a_i to the range of real numbers in [-3, +3], we trained a 3-3-1 multi-layer perceptron neural network to find the value of y for every possible combination of cardinal a_i coefficients. Then, we extracted a vector of size 16, being the number of weights and thresholds of each of these networks, normalised with respect to the length of each pattern vector. These normalised vectors form the 343 input patterns for training our analyser network.

The size of analyser network for this example was 16-5-3. Once such a network is trained the extracted weight vector of a network to be analysed can be input. The three output nodes of the analyser network will find the a_i coefficients for the particular equation of your network.

We did a number of experiments, for producing the 343 trained ANNs from which to extract weight vectors to use in training the analyser network. Each of the 343 networks was trained using back-propagation to learn a particular equation, by inputting a training set of size 1331 (11*11*11)

generated to cover the possible range of input patterns by steps of 0.1 for each variable. The program to train ANNs of size 3-3-1 and 3-4-1 took 37 and 29 hours run time on DEC machines to prepare weight matrices for training sets for our experiments.

We had no difficulty in training our analyser neural network and could analyse some simple networks immediately. We noted the complexity of the task in handling systems with a lot of variables, due to the many possibilities to adjust weights in very different forms (depending on weight initialisation, learning rate, momentum, overall error, number of epochs, and so on). This complexity was especially apparent in more extended domains. Thus we found we needed to do some pre-processing on the weight matrices before inputting into the analyser neural network.

This need lead to the following experiments in analysing the weight matrix of trained neural network s, which resulted in our hidden index approach.

3 Hidden Indexes

One possible way to prepare the raw weight matrix of a trained neural network as input to the analyser neural network is to multiply the weight matrix (M) by its transpose (MT) which results in a symmetric matrix (C), see Figure 3. At the first stage the experiments were done on a one hidden layer neural network with three hidden units. The experiments were repeated on neural networks with different numbers of hidden units, and the method worked consistently.

$$M = \begin{vmatrix} W_{11} & W_{11} & \cdot\cdot & W_{11} \\ \vdots & \vdots & \vdots\vdots & \vdots \\ W_{n1} & W_{n2} & \cdot\cdot & W_{nm} \\ th_1 & th_2 & \cdot\cdot & th_m \end{vmatrix} = \begin{vmatrix} V_1 \\ \vdots \\ V_n \\ V_{n+1} \end{vmatrix} \qquad C = \begin{vmatrix} V_1 V_1 & V_1 V_2 & \cdots & V_1 V_{n+1} \\ V_2 V_1 & V_2 V_2 & & V_1 V_{n+1} \\ \vdots & \vdots & \vdots & \vdots \\ V_{n+1} V_1 & V_{n+1} V_2 & \cdots & V_1 V_{n+1} \end{vmatrix}$$

Figure 3. Weight matrices

It appeared from considering the result matrix that each column of the matrix shows the magnitude of coefficients of input variables and measures the amount of effect each input unit has on the output variables. We call these coefficients *hidden indexes*. A comparison of correctness of hidden and causal indexes for the 343 different ANNs is shown in Table 1, each column measures the correctness of the found coefficients by the Euclidean estimation criterion.

Table 1. Correctness of found coefficients by the Euclidean estimation criterion

Err. Cr.	HI@1	HI@2	HI@3	HI@T	AVG	HI@avg	differ.	CI
0.02	74	36	40	56	25	25	26	23
	5.91	10.33	16.23	1.49	7.44	8.22	2.71	2.74
0.002	18.14	11.08	24.58	2.19	2.19	9.6	1.04	1.26
0.00075	6.99	4.23	4.52	0.81	5.88	3.36	1.20	1.43

HI@1 to HI@3 show Hidden Indexes found by input weights, HI@T shows the indexes by thresholds and HI@avg contains average of indexes which are calculated by HI@1 to HI@3. The *differ.* column contains HI@avg - HI@T.

For a better understanding of these phenomenae, we can compare the convergence curves in ANN, CI, and HI. These are shown in Figures 4 and 5.

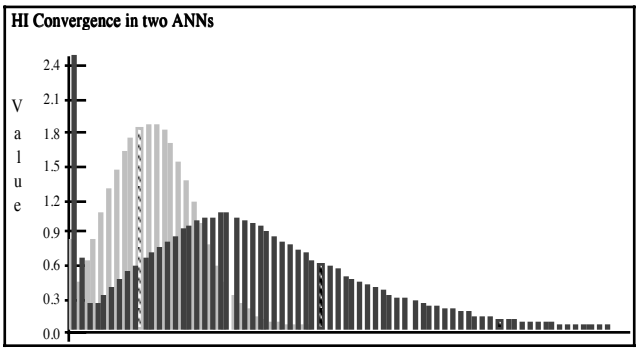

Figure 4. Comparing HI convergence of different ANNs.

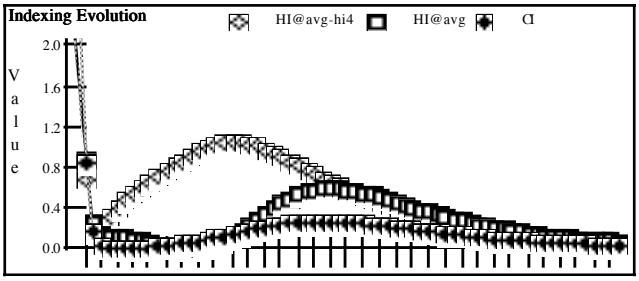

Figure 5. Comparing HI, CI convergence curves.

4 A Real World Example

In figure 6 we compare the obtained significance coefficients of input variables in a network trained to estimate the final mark of students by four methods: causal index, hidden index, sensitivity analysis, and multivariate linear regression. For hidden indexes we have calculated indexes by HI@avg - HI@4.

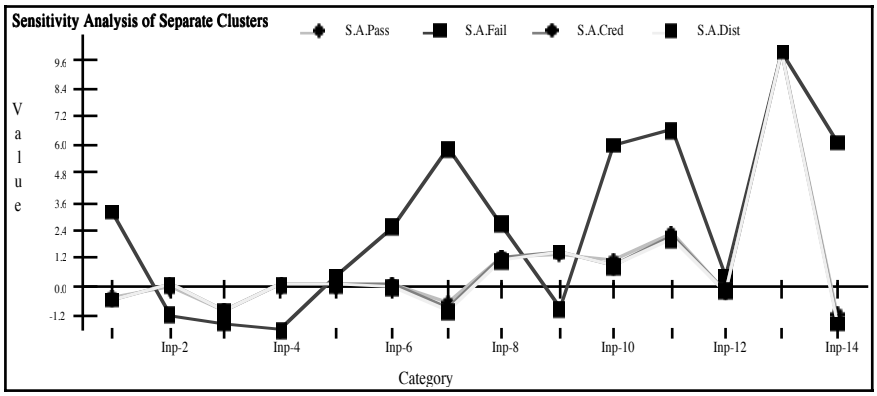

Figure 6. First order Sensitivity Analysis on each input of a trained ANN, with all other inputs held constant at canonical of 4 separate clusters.

We argue that in spite of the differences in obtained coefficients, all of them are acceptable, the benefits of our method are described below.

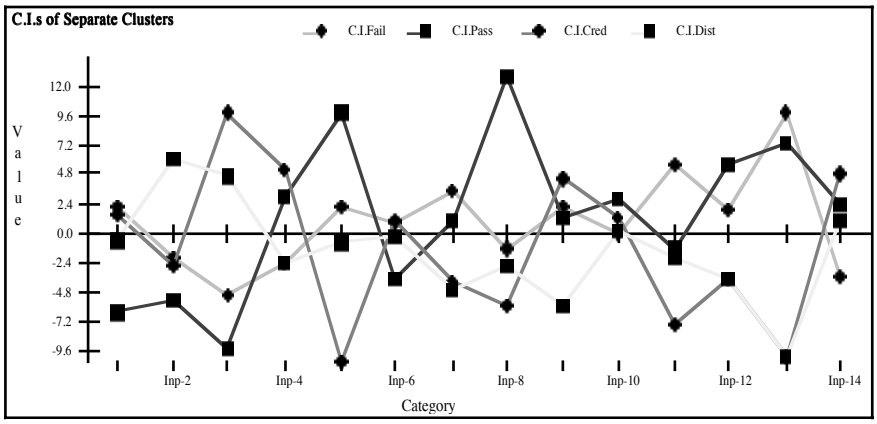

Figure 7. Causal index method on inputs of trained ANN for 4 clusters.

Back-propagation trained ANNs can be considered as non-parametric models which can adapt network weights to estimate any function [15]. They can

be compared with multivariate multiple nonlinear regression [16]. In a multivariate linear regression, entering a new variable into a system can have significant effects on the coefficients of previously entered independent variables or may fail the T-significance test hypothesis of a desired significance level. Adding insignificant variables will not increase the explanatory power of the equation, and will lead to the loss of one degree of freedom.

One way of computing sensitivity analysis (SA) is to perturb one input slightly with other inputs held constant at 0, 0.5, or 1. This approach is inefficient because SA usually leads to completely different results in separate regions. Figures 7 and 8 show the results of CI and SA for four separate classes.

Another way of computing sensitivity analysis is to compute SA for one input over the range of values for one or a few other inputs. But this method is also inefficient because having a separate perturbation for each input over its range is time consuming and makes the analysis of the result difficult.

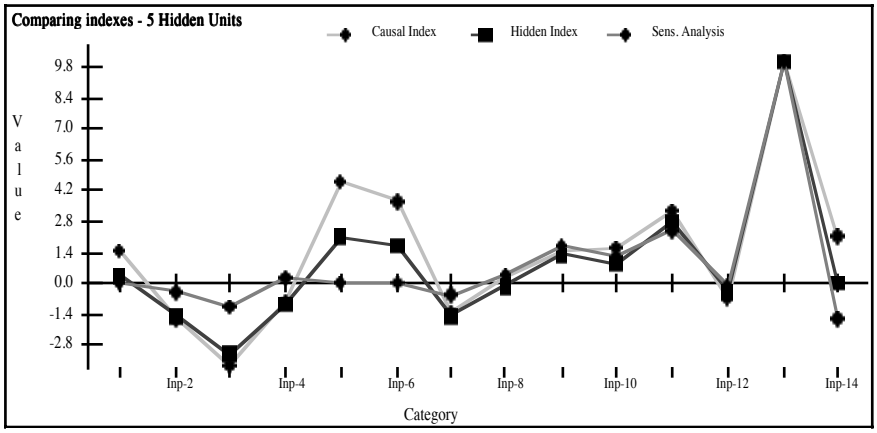

Figure 8. Hidden Indexes vs CIs and SA approaches applied on a trained neural network with 5 hidden units to predict student final marks.

As one appropriate solution to this problem we have used the canonical of each separate cluster which we obtained by training a Bidirectional Neural Network [17]. We also averaged the effects of two different positive and negative dithers. We show in Figures 9 & 10 that results of HI and SA are less dependent on structure of ANNs than CI, and HI results are usually closer to SA than CI.

In another experiment we trained networks to learn the relation between three dependent variables. We saw that if input variables are not mutually exclusive then no pattern was seen on the coefficient of independent variables and they will change in each training regime (eg. for different learning rate

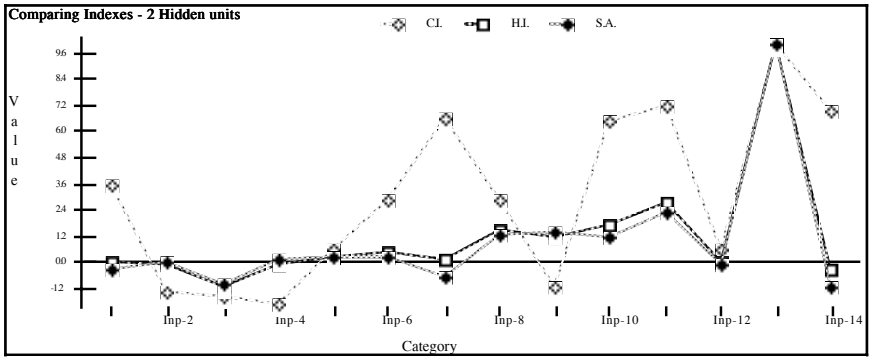

Figure 9. Comparing Hidden Indexes vs CIs and SA approaches applied on a trained neural network with 2 hidden units.

parameters). This may be compared with some problems of multi-collinearity in statistics with intercorrelations between supposedly totally independent variables.

5 Conclusion

We trained supervised error back propagation neural networks as analyser neural networks. Using this technique it is possible to develop a toolkit of analyser networks trained for recognising linear and non-linear regressions. The process of discovering the rules obeyed by a specified network is very quick. Since the analyser networks are already trained, we only need to input the normalised weight and bias vector of the specified network, and then propagate through to produce a result which can be read off directly.

There is little regularity in the raw weight matrix of ANNs and this would make it very difficult to analyse, whether by our analyser neural nets or other techniques. However, we have shown that there are underlying regularities which govern weight distributions in ANNs, and extracting these regularities using our technique of hidden indexes is much more effective than using raw weight matrixes in analysing ANNs. In other work not reported here, comparing the results of different approaches showed that they may also be used in determining the optimal number of epochs as well as optimal number of hidden units in order to have more robust networks. There is still a lot of work to be done on using these regularities in optimising the structure and training methods.

References

[1] Gallant, SI "Connectionist expert systems," *Communications of the ACM*, vol. 31, no. 2, pp. 152-169, February 1988.

[2] Gallant, SI *Neural Network Learning and Expert Systems*, MIT press,

1993.

[3] Bochereau, L and Bourgine, P “Expert systems made with neural networks,” *Int. Joint Conf. on Neural Nets*, vol. 2, pp. 579-582, 1990.

[4] Saito, K and Nakano, R " Automatic Extraction of Classification Rules," *International Neural Network Conference*, Paris, pp 9-13, July 1990.

[5] Towell, G, and Shavlik, JW “Extracting refined rules from knowledge-based neural networks,” *Machine Learning*, 13(1), pp. 71-101, 1993.

[6] Gedeon, TD and Turner, H “Explaining student grades predicted by a neural network,” *Proc. Int. Joint Conf. Neural Networks*, 609-612, Nagoya, 1993.

[7] Breiman, L, Friedman, J, Olshen, R and Stone, C *Classification and Regression Trees*, Monterrey, Caiwadsworth, 1984.

[8] Quinlan, JR "Learning With Continuous Classes," *Proceedings of 5th Australian Joint Conference on Artificial Intelligence*, 1992.

[9] Hinton, GE "How Neural Networks Learn from Experience," *Scientific American*, pp. 105, Sep, 1992.

[10] Good, RP and Gedeon, TD "Network Analysis Techniques as visualisation tools," *Proc. Int. Conf. on Non-Linear Theory*, pp. 945-952, Hawaii, 1993.

[11] Pratt, L, Nicodemus, S "Case studies in the use of a hyperplane animator for neural network research," *Proc. of the IEEE Int. Conf. on Neural Networks*, vol. 1, pp.78-83, 1994.

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

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

[14] Hashem, S "Sensitivity Analysis for Feedforward Artificial Neural networks with Differentiable Activation Functions," *IEEE*, 1, pp 419-424, 1992.

[15] White, H *Artificial Neural Networks: Approximation and Learning Theory*, Oxford, UK: Blackwell, 1992.

[16] Sarle, WS "Neural Networks and Statistical Models," *Proceedings of the Nineteenth ANNual SAS Users Group Internat. Conference*, April, 1994.

[17] Nejad, AF and Gedeon, TD “Bidirectional MLP Neural Networks and Their Applications,” *Proc. Int. Symp. Artificial Neural Networks*, Taiwan, 1994.