EXTRACTING CONTEXTUAL IF-THEN RULES FROM A FEEDFORWARD NEURAL NETWORK

Tamás D. GEDEON and **Harry S. TURNER**
School of Computer Science & Engineering
The University of New South Wales
P.O. Box 1, Kensington 2033
AUSTRALIA

Abstract

Neural networks can be trained to provide solutions in application domains where clear rules which would allow symbolic solutions do not exist. Neural networks in these domains still suffer from a major disadvantage, in that there is no explanation for why a particular decision was made by the network.

We present our method of generating if-then rules expressing the trained neural network's behaviour. By using the causal index on characteristic input patterns, we produce a list of inputs which were significant in reaching the decision made, a set of rules governing this decision, and the next most likely decision the network could have made. This method correctly produced rules for 94% of the decisions made by a sample network.

1. Introduction

Feed-forward networks of a few layers trained by back-propagation can be used to solve a variety of problems. In this paper we will generally assume a feed-forward network of three layers of processing units. All connections are from units in one level to the subsequent one, with no lateral, backward or multilayer connections. Each unit has a simple weighted connection from each unit in the layer above. The network is trained using a training set of patterns with desired outputs, using back-propagation of error measures [1]. Most workers now use batch rather than sequential pattern by pattern updating of weights. In the examples we cite here we have used the basic logistic activation function $y=(1+e^{-x})^{-1}$. Training by back-propagation is popular because of its simplicity theoretically, and the ease of use and production of such networks, and the wide availability of low cost simulators.

The ability to learn from examples makes these neural networks useful and powerful tools. One application that is becoming increasingly popular is that of using the trained neural network in the role of a human expert. This has the advantage of reducing large amounts of expensive expert time by learning from example data during training. This method has obvious advantages, however unlike conventional expert systems the neural network has no ability to provide any rule trace, or some sort of explanation as to how it comes to its conclusions. Rather than storing sets of rules the knowledge contained in a neural network is stored in the weight values distributed throughout the entire network.

We have produced an explanation facility for back-propagation trained neural networks [2, 3]. Explanations do not simply consist of a set of rules (or rule traces) as to why the network came to its conclusion, but also include identification of important factors in the input, and the next most likely output of the network.

2. Causal Index

The Causal Index C_{ki} is the rate of change of k with respect to i. This indicates the relationship between the k^{th} output and the i^{th} input neurons in the trained neural network. The value of C_{ki} indicates a positive or negative correlation between input and output signals. Using this value, explanation of the importance of each input could be given by using equations such as: If C_{ki} is Positive and Large then 'If i is large then k is large'.

Consider the three layer network described in Figure 1. The outputs of the neurons in the network are given by the formulae:

$$y_k = f(U_{k2}) = (1 + e^{-U_{k2}})^{-1}$$
$$h_j = f(U_{j1}) = (1 + e^{-U_{j1}})^{-1}$$
$$U_{k2} = \sum_j w_{jk} h_j$$
$$U_{j1} = \sum_i w_{ij} x_i$$

Where

- U_{in} : Sum i inputs in n^{th} layer,
- w_{ij} : Weight from neuron i to neuron j,
- y_k : Output neuron,
- f : Sigmoid function, and

- h_j : Hidden neuron.

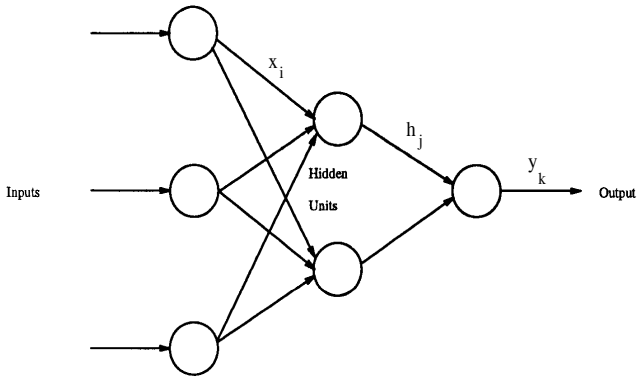

Figure 1: Structure of a three layered ANN with one output

The rate of change of an output neuron y_k with respect to an input neuron x_i is found by calculating the derivative dy_k/dx_i using the chain rule of differentiation.

$$\frac{dy_k}{dx_i} = \frac{dy_k}{dU_{k2}} \cdot \frac{dU_{k2}}{dh_j} \cdot \frac{dh_j}{dU_{j1}} \cdot \frac{dU_{j1}}{dx_i}$$

$$= f'(U_{k2}).f'(U_{j1}).\sum_j w_{jk}.w_{ij} = C_{ki}$$

This method has also been used in [4, 5], using the assumption that the product $f'(U_{k2}).f'(U_{j1})$ is constant for all k and j . The influence of x_i on y_k could thus statically be determined from the weight matrix of the trained network.

While this assumption may hold in some domains, we have found that this does not hold in the domains we have tried. Figure 2 demonstrates the value of the product $f'(U_{k2}).f'(U_{j1})$in a simple four input, two hidden unit, and one output network, by holding three of the inputs constant, and varying the fourth from *0* to *1*. As we can readily see, the value varies, and is not close to a constant.

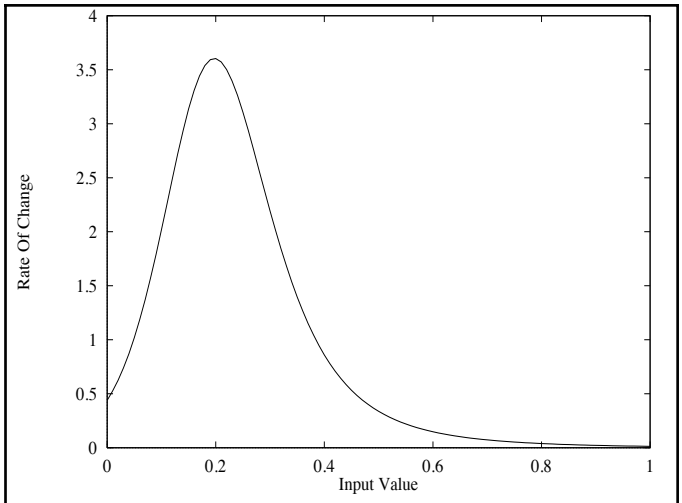

Figure 2: All inputs initially OFF

To produce a general solution to the task of justifying and explaining conclusions made by trained neural networks, the Causal Index relationship of inputs to outputs needs to be interpreted it to produce accurate, understandable explanations which describe key factors and their relationships. This interpretation of the Causal Index information is using *Characteristic patterns*. We have examined simple networks in which the functionality is known, then generalised to other networks and tested for accuracy. This paper includes an extended example using real data. A number of approaches not depending on the Causal Index have been proposed, for example [6-9]. The common problems most of these methods suffer from is either a lack of generality, or an explosion of generated rules.

3. Characteristic Patterns

The formula used in calculating the Causal Index causes one major problem: the results are input specific. This is a problem in two ways. If the analysis can not be generalised to satisfy any set of possible inputs or even any arbitrary subset of the set of possible inputs, then creating explanations for each input pattern is firstly not very efficient, and secondly such extremely specific explanations are not ideal. That is, an explanation is at least implicitly something humans can generalise from. Thus, we require a more general explanation which focuses on the key elements distinguishing particular input patterns.

This problem has been overcome by using input values representative of the input set. Finding a single input pattern that is representative of the entire input set is impossible. To achieve this an input pattern must be found representing all the patterns that cause an output to be both on and off; an obvious contradiction. To solve this problem input patterns are split into classes according to their effect on an output. When the input pattern causes the output being analysed to be turned on, it is classed as an *ON*

input pattern, otherwise it is classed as an *OFF* input pattern.

Using the mean or median of each input value, a pattern representing each input class is created as a *Characteristic* pattern for that class. A characteristic *ON* pattern is a pattern characteristic of those input patterns which turn an output on. Similarly a characteristic *OFF* pattern is a pattern characteristic of those input patterns which turn an output off. We had used the median value initially, expecting to use a more statistically mature means of finding the *Characteristic* patterns, however in all the examples we have examined so far, this has not yet been necessary. Note that we consider here networks where the output units represent membership in categories by high output values. For networks where an output unit's value is used directly and not just as a threshold, we can define *Characteristic* patterns over subranges of the value of the output unit activation.

The *Characteristic* patterns for a number of simple 4-2-1 networks trained on boolean functions of the inputs, are shown below. Where either the *ON* or *OFF* pattern is not useful, it has generally been omitted. This occurs for instance in the network trained on the disjunction of four boolean inputs with the *Characteristic ON* pattern, where modifying one input when the network produces an *ON* output has no significant effect.

Pattern - Network	Input 1	Input 2	Input 3	Input 4
ON - Conjunction	1.0	1.0	1.0	1.0
OFF - Conjunction	0.466	0.466	0.466	0.466
OFF - Disjunction	0.0	0.0	0.0	0.0
OFF - Conj of Disj	0.284	0.284	0.284	0.284
ON - Disj of Conj	0.7	0.7	0.7	0.7
OFF - Disj of Conj	0.3	0.3	0.3	0.3

Table 1: Characteristic patterns for a number of networks

4. Deriving logical formulae

We show here the derivation of logical formulae from Causal Index graphs of trained feed-forward neural networks, in terms of a number of simple networks in which the functionality is known. This is an expository device, the networks demonstrated are not considered significant in themselves.

4.1. Logical Conjunction

Figure 3 shows the result of calculating the Causal Index for each of the four inputs in the *ON* pattern.

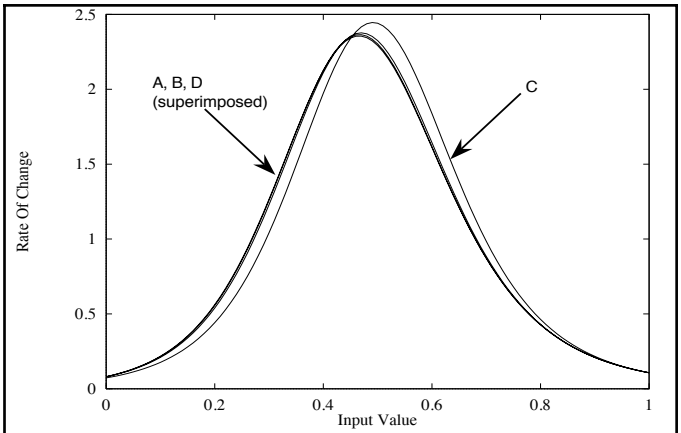

Figure 3: Causal Index of inputs for (A AND B AND C AND D) network – Ch. ON

The four separate curves shown in this graph are almost identical. In each case, the network's result is completely dependent on the presence of all of the inputs. We know that the result produced by the network is the logical conjunction of the four inputs. Inputs therefore, that produce curves as shown in Figure 3 that result in complete change of the networks output (ie from off to on) are interpreted as crucial inputs. Any crucial inputs are combined using the conjunction operator.

The Causal Index was also calculated for each of the four inputs in the characteristic *OFF* (input) pattern for a network trained to perform the logical conjunction of four inputs. The graph of this calculation is not shown, as it indicates a consistently small value of the Causal Index. Changing any single input therefore has no effect on the network's output. The *Characteristic* patterns for all other cases which are similarly not useful have been omitted from Table 1.

4.2. Logical disjunction

Another network was trained to recognise the logical disjunction of four inputs, and the Causal Index for each input in the *OFF* pattern (Table 1) calculated.

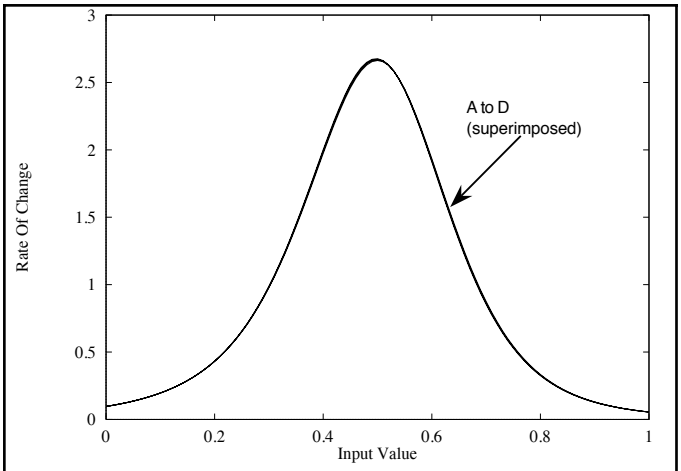

Figure 4: Causal Index of inputs for (A OR B OR C OR D) network – Ch. OFF

The graph in Figure 4 shows all four curves are almost identical. The Causal Index calculated for each input in the characteristic ON input shows negligible change and is therefore omitted.

We know that this network performs the logical disjunction of four inputs. The only major difference between the results in Figure 3 and Figure 4 is that the curves are produced using a *Characteristic ON* input pattern in the first and a *Characteristic OFF* pattern in the second. Note that none of the inputs are (singly) crucial for the output to become *on*. Instead, any one of the inputs that produce the curves in Figure 4 will turn the output from *off* to *on*. Curves such as these found in Figure 4 will therefore be interpreted as the logical disjunction of the inputs from which they are derived.

4.3. Conjunction of Disjunctions

To examine the conjunction of disjunctions a network was trained to recognise the logical formula *((A OR B) AND (C OR D))*. The *Characteristic OFF* input for the network (from Table 1) yields the graph of Causal Index shown in Figure 5.

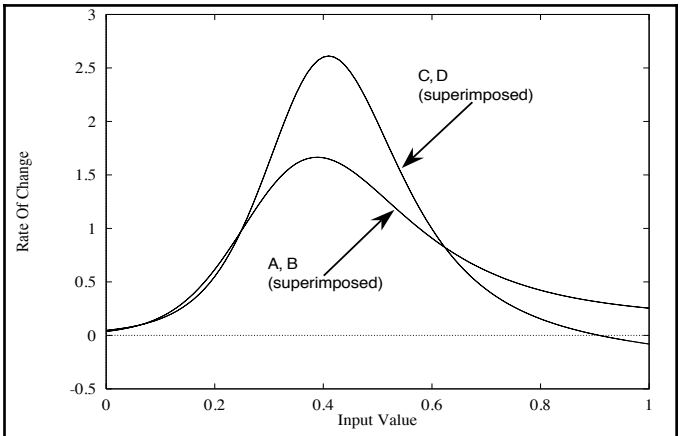

Figure 5: Causal Index of inputs for ((A OR B) AND (C OR D)) network – Ch. OFF

Note that the peaks in the rate of change of the output in this case do not indicate a complete change of network output (this is found by examining the network's output over the same change in input). Thus, change in any single input in this pattern does not turn the output neuron *on*. The inputs *A* and *B* are represented by the smaller two curves (appearing identical), and *C* and *D* by the larger two curves (also appearing identical). Using the previously devised interpretations, Figure 5 indicates that turning any two of these inputs on at the same time will cause the output to change from *off* to *on*. The logical formula that would be produced under this interpretation from the graph is: *((A OR B) AND (C OR D)) OR ((A OR C) AND (B OR D)) OR ((A OR D) AND (B OR C)))*. This logical formula allows the output to be *on* in more cases than the network produces, hence some limitation in the interpretation must be found.

The sum of the maximum rate of change of Inputs *A* and *C* and a similar sum of Inputs *A* and *D* are well nigh identical, likewise the sums of Input *B* and *C*, and Inputs *B* and *D*. These are the only combinations that result in the sum of the maximum Rate of Change (or Causal Index value) being the same. This is desired as we know for this network that all of the inputs are equally important, and therefore should have the same effect (hence rate of change) on the output. For this reason the interpretation of such a graph is that multiple inputs not turning the output *on*, but appearing as peaks in a graph of a *Characteristic OFF* pattern are combined in pairs of equal (or very similar) maximum rates of change using the *OR* connector. These pairs are then

combined using the *AND* connector.

4.4. Disjunction of Conjunctions

The final logical formula that we need to consider is *((A AND B) OR (C AND D))*. The *Characteristic ON* input (from Table 1) was used to produce the graph of the Causal Index for each input in Figure 6.

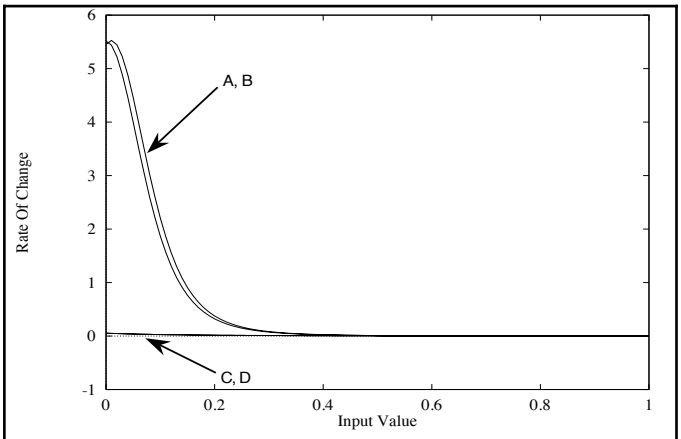

Figure 6: Causal Index of inputs for ((A AND B) OR (C AND D)) network – Ch. ON

This shows large change in two of the inputs, *A* and *B*, and no change in the other of the inputs *C* and *D*. Using previous interpretations, the logical formula *(A AND B)* can be deduced from this graph. In this case, the Causal Index graph for the *Characteristic OFF* pattern also shows activity, see Figure 7.

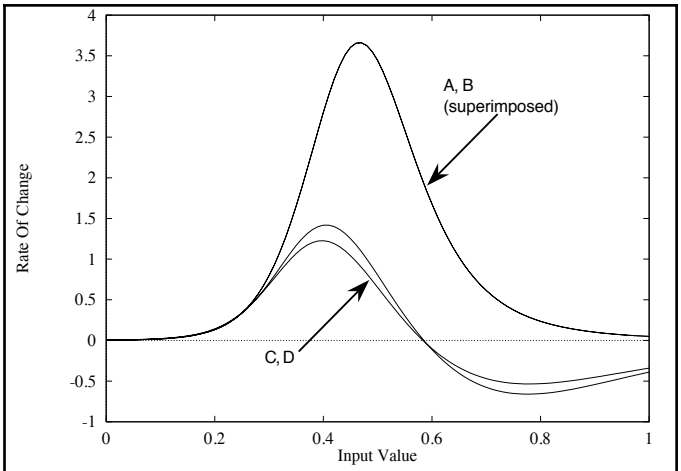

Figure 7: Causal Index of input for ((A AND B) OR (C AND D)) network – Ch. OFF

This shows that inputs *A* and *B* have identical high peaks, and that inputs *C* and *D* have smaller similar peaks. The interpretation from this graph is that the output of the network is turned *on* when either *A* or *B* is *on*, however we already know from Figure 6 that we require both *A* and *B* to be *on*. Although showing small peaks, the output is not turned *on* by singly altering either of the inputs *C* or *D*. The interpretation from the section on disjunctions needs to be extended. There, inputs with similar low peaks in the Causal Index were combined in pairs using *OR* before being combined using *AND*. Here there are only two inputs, and are combined using *AND* only. In this case, the output changes to *on* when both *C* and *D* are *on*. The interpretation from this graph is thus *(A OR B OR (C AND D))*, which is combined with the results from the other *Characteristic* pattern.

Combining the interpretation is straightforward, producing the logical expression stating the constraints required to produce a positive output from

the network is *((A AND B) OR (C AND D))*, which is the desired result. Note that we have achieved the correct result, even though some asymmetry in the final weights of the trained network produced some difference in the apparent significance of the four inputs. This is shown by the difference in behaviour of *A* and *B* versus *C* and *D*. Since the trained network performed correctly notwithstanding this asymmetry, any rule derivation system must be able to handle this to be useful in practice. Our method is clearly sufficiently robust.

4.5. Exclusive OR

It should be noted that the logical formulae described above are complete, all logical expressions can be expressed in these terms. For example *A XOR B* can be expressed in the form *(A AND ~B) OR (~A AND B)*, which is a disjunction of two conjunctions.

Nevertheless, since the exclusive OR problem is traditionally connected to the back-propagation network, we will demonstrate that our method can extract the appropriate logical rule. A network consisting of two inputs, two hidden units and one output was trained to recognise *A XOR B*.

This network is different to the previous ones, in that the *Characteristic ON* and *OFF* patterns are identical, having the value *0.5* for both of the inputs.

Figure 8 shows the Causal Index graph for the XOR network.

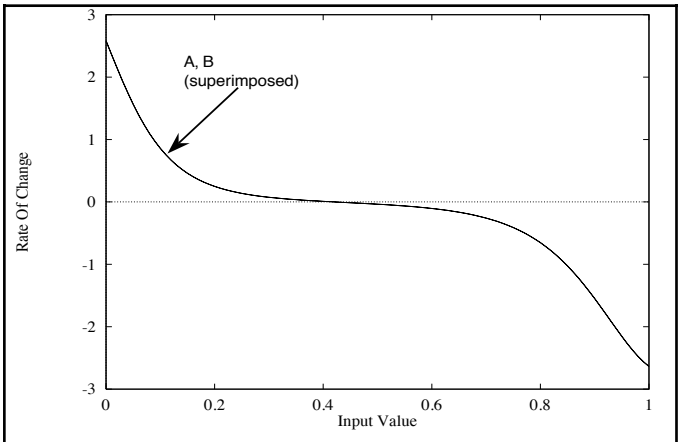

Figure 8: Causal Index of inputs for (A XOR B) network – both Ch. ON & Ch. OFF

The two curves are identical. Two peaks are present for both input *A* and input *B*, one positive and one negative. The positive peak on low value of both inputs on the *Characteristic ON* pattern indicate that a low value on either one of them is enough to turn the output *on*. This produces the interpretation *(A OR B)*. The peaks on the *Characteristic OFF* pattern would produce *A AND B*, however the peaks are negative, thus the interpretation becomes *~(A AND B)*. Simple algebra on *(A OR B) AND ~(A AND B)* produces the desired *(A AND ~B) OR (~A AND B)*.

4.6. Formalisation of Interpretation

For clarity, the interpretations derived in the preceding sections are stated clearly below:

- Inputs having a large peak in rate of change of an output (ie at least enough to turn the output *on*) in a *Characteristic ON* input pattern are combined using the conjunction operator. Eg, *A AND B*.
- Inputs having a large peak in rate of change of an output (ie at least enough to turn the output *on*) in a *Characteristic OFF* pattern are combined using the disjunction operator.Eg, *A OR B*.
- Two inputs that appear on the graph of a *Characteristic ON* input pattern that do not turn the output completely *off* if set to zero are interpreted as the disjunction of these inputs. Eg, *A OR B*.
- Two inputs that appear on the graph of a *Characteristic OFF* input pattern that do not turn the output completely *on* when set to one, are interpreted as the conjunction of these inputs. Eg, *A AND B*.
- As a further restriction in the above two interpretations, if more than two inputs occur that do not completely turn an output *on* or *off*, the combinations are done by firstly combining inputs with the most similar graphs with the *AND* or *OR* connectors respectively, then these grouped inputs are combined by the apposite connector. Eg, *(A OR B) AND (C OR D)* versus *(A AND B) OR (C AND D)*.
- A negative peak in any of the above signifies *NOT*.
- As with all logical formulae, conjunctions take precedence over disjunctions.

5. Deriving numerical formulae

We show here the derivation of numerical formulae from Causal Index graphs. These formulae allow non-Boolean values of inputs. The extension is demonstrated in terms of a simple network composed of two inputs, two hidden units, and one output unit trained on the formula *((A < 0.2) OR (B > 0.6))*.

The Causal Index graph for each input of the *Characteristic OFF* pattern is shown in Figure 9.

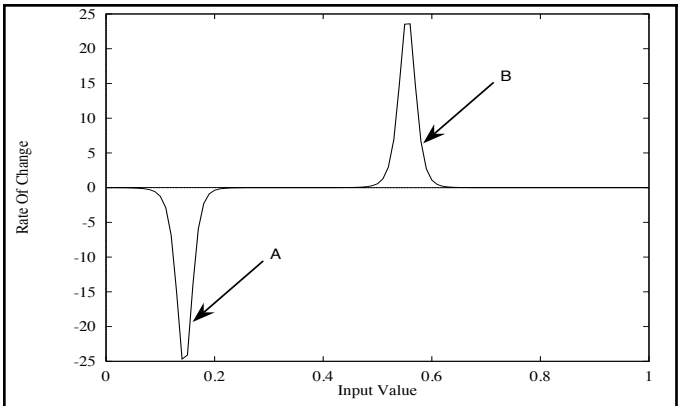

Figure 9: Causal Index of inputs for ((A < 0.2) OR (B > 0.6)) network – Ch. OFF

The graph shows a large negative peak in the rate of change for input *A* which then returns to a constant level of zero at *0.2*, and a large positive peak in the rate of change for Input B which returns to zero at *0.6*. This leads to the conclusion that the point at which the Rate of Change returns to a constant level is the value the input must reach to be significant. Since the graph is that of a *Characteristic OFF* pattern, the previous formalisation leads us to the formula *((A < 0.2) OR (B ≥0.6))* for the output to be *on*. This is the desired conclusion. We could have also used the negative peak to indicate *~(A ≥ 0.2)*, which translates to *(A < 0.2)*. Note that the term *(B ≥ 0.6)* is essentially the same as *(B > 0.6)* at the numerical resolution to which this neural network was trained.

Thus, when dealing with real input values, the cut off point is the value at which the rate of change begins to stabilise, usually when it returns to zero. This point can then be used for numerical interpretation using the formalised interpretation previously discussed for the inputs' relationships with the output.

5.1. Formalisation of Interpretation

The additional interpretations for numeric formulae.

- Inputs having a large positive peak in rate of change of an output (ie at least enough to turn the output *on*) use the ≥ comparison operator, with a value derived from the point where the rate of change stabilises. Eg, *(B ≥ 0.6)*.
- Inputs having a large negative peak in rate of change of an output use the < comparison operator. Eg, *(A < 0.2)*.
- The above interpretations apply equally to *Characteristic ON* and *Characteristic OFF* patterns.

6. Deriving rules in a real example: the Mark Predictor network

The network analysed is a three layered neural network consisting of fourteen inputs, five hidden units and four output units. The network has been trained on a set of assessment marks in an undergraduate Computer Science subject to predict the final result that a student will receive [2, 3]. The assessments shown are combined to yield 40% of the total mark, the last 60% being the exam which is omitted. Thus, the task is to predict the final mark which was calculated using the exam mark, from only the 40% of class assessments during the teaching session.

The classification of marks fall into the following categories:

- Distinction or above, being a mark of 75 or greater, represented by output 1.
- Credit, being a mark between 65 and 74, represented by output 2.
- Pass, being a mark between 50 and 64, represented by output 3.
- Fail, being a mark less than 50, represented by output 4.

After training, the network classified the training set with an accuracy of 96%, and the test set with an accuracy of 75%. Training was stopped to avoid overtraining when the error on the test set began to increase, which indicates the point of maximum generalisation. For the rest of our discussion, the network's output is assumed to always be correct. This is appropriate since we are providing explanations for the conclusions reached. It is possible that explanations for conclusions on a test set could be used to deduce whether or where the network has learnt incorrectly. Here we will concentrate on providing explanations for the conclusions reached on the training set.

The *Characteristic ON* patterns for this network were derived from the training set in the manner described previously. The set of these patterns will be denoted *CON* subsequently, using a subscript representing the output concerned. Thus, CON_{Fail} is the *Characteristic ON* input pattern for the *Fail* output unit.

The *Characteristic OFF* patterns can not be derived as simply as described previously. Simply averaging the inputs that do not cause an *on* output is not an accurate way of finding the *Characteristic OFF* pattern. This example network is a very good illustration of this problem. When trying to find the *Characteristic OFF* pattern for a *Pass* output, the non-*Pass* input patterns such as those classified as *Distinction*s, *Credit*s and *Fail*s are averaged producing an intermediate pattern. This intermediate pattern when input to the trained network was classified as a *Pass*, and thus is not very useful as a

Characteristic OFF pattern for the *Pass* output. Since in this case we know that at most one output unit will be on for any particular input pattern, we can use the *Characteristic ON* patterns of the other output units as the *Characteristic OFF* patterns for this output unit. The set of these patterns will be denoted *COFF*, using a subscript representing the output concerned, and a superscript denoting another output unit whose *CON* is being used. Thus, $COFF_{Fail}^{Pass}$ denotes the *Characteristic OFF* pattern for the *Fail* output which is made up of the *Characteristic ON* pattern of the *Pass* output.

In the following section we will derive the rules for the *Distinction* output class.

6.1. Distinction output

The rules derived by our method are always in the context of the input pattern used, and the output decision. Thus, rule sets in this section are interpreted as "IF <rule set> THEN Distinction". In the discussion in sections 4 and 5, the networks were sufficiently simple, that this point was left implicit. In the rest of this section, we will derive the rules in all of the relevant contexts for the Distinction decision made by the trained feed-forward network.

The *Characteristic ON* patterns for each of the outputs is shown in Table 2.

	CON_{Dist}	CON_{Cred}	CON_{Pass}	CON_{Fail}
Crs	0.7	0	0.65	0.35
Stg	1	0.5	0.85	1
Enr	1	1	0.98	1
Tutgp	0.5	0.75	0.67	0.61
lab2	0.7	0.85	0.52	0.4
TutAss	0.4	0.7	0.44	0.4
lab4	1	0.85	0.52	0
H1	0.8	0.8	0.67	0.4
H2	0.8	0.8	0.65	0
lab7	0.7	0.55	0.5	0
P1	1	0.7	0.52	0
F1	0.7	0.3	0.42	0
Mid	0.72	0.59	0.39	0.21
lab10	0.5	0.5	0.49	0

Table 2: Characteristic ON pattern for all outputs

6.1.1. Distinction result with input pattern similar to CON_{Dist}

The simplest case occurs when the input pattern is most similar to the Characteristic ON pattern for the output. That is, firstly the trained network on a particular input pattern decides that the categorisation is Distinction by turning that output unit on, and secondly that input pattern is most similar to the Distinction Characteristic ON pattern. Thus, the input pattern looks like a classic Distinction case, and is categorised as such by the trained network.

The Causal Index graph for the CON_{Dist} pattern is shown in Figure 10.

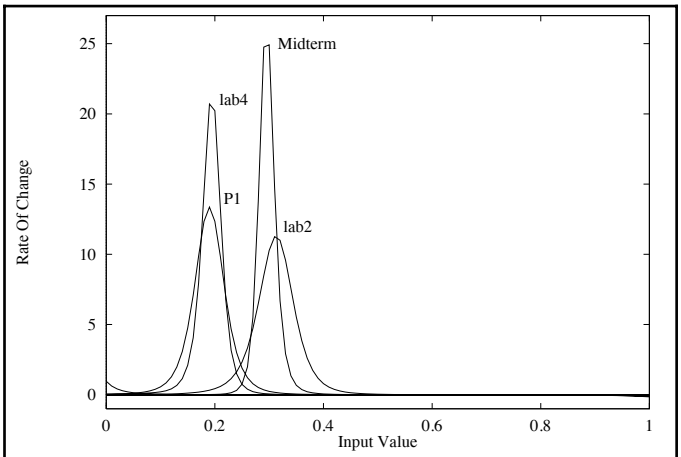

Figure 10: Causal Index of inputs for Mark Predictor network – CON_{Dist}

The four positive peaks in the graph produce simple numerical rules, combined using *AND* operators, as discussed previously. These rules derived from this pattern are shown below, in Table 3.

Character. Pattern	Rule Set
CON_{Dist}	(lab2 ≥ 0.44) AND (lab4 ≥ 0.23) AND (P1 ≥ 0.27) AND (Midterm ≥ 0.37)

Table 3: Rules produced from the Causal Index graph for Mark Predictor network – CON_{Dist}

This is the simplest case of rule generation, for the *Characteristic ON* pattern matching the network decision. The interpretation of these rules in the context of the example needs some discussion. The rules would be presented to the user after presenting the most similar *Characteristic* pattern. As indicated, that pattern was the CON_{Dist}, from Table 2.

The rules in Table 3 indicate the performance required so as to still match the *Distinction* archetype respresented by CON_{Dist}. For example, so long as the second laboratory assessment (*lab2*) mark is not below *0.44*, the result will be a *Distinction* mark for a particular input pattern representing a particular student's performance. This explains the quite low values of the inputs in the rule set compared with the values in the *Characteristic* pattern.

The *Characteristic* pattern indicates that fairly high marks are required overall to achieve a *Distinction* result. The four inputs identified in the rule set in

Table 3 are very plausible to be so significant. There are two possible explanations for this, firstly the weighting of these four in the final grade is high, or secondly, the material covered in those four assessments is representative of the material in the final exam. While the weighting of *Midterm* is high, it is also clearly representative of the material in the final examination. The other three inputs identified as important are not weighted particularly high in the calculation of the final grade. Lack of a good level of performance during laboratory exercises and the procedural programming assignment plausibly shows a deficiency leading to a less than *Distinction* performance. A major strand of the subject is on functional programming, and thus it is initially surprising that the assignment *F1* does not occur in the rule set. This strand is taught at a more introductory level, and may be either sufficiently well understood in general, or not understood at all, that a particularly low performance in the assignment does not modify the final grade.

6.1.2. Distinction result with input pattern similar to CON_{Cred}

In this case we would assume that the most similar *Characteristic* pattern is not the *Distinction* pattern. However, the trained network on the particular input pattern decides that the categorisation is *Distinction* by turning that output unit on. Thus, the input pattern looks like a classic *Credit* case, but is categorised as a *Distinction* by the trained network. This means we must look at the $COFF_{Dist}^{Cred}$ Causal Index graph. That is, we use the *Characteristic ON* pattern for a *Credit* as an *OFF* pattern for the *Distinction* output.

In the Mark Predictor network example, there are no cases of this nature. All patterns in our set which are most similar to the *Credit* mark's *Characteristic ON* pattern are actually either categorised as a *Credit* mark by the trained network, or the pattern satisfies the unmodified rules for a *Distinction* result derived from the CON_{Dist} pattern. The Causal Index graph is not shown because it is essentially featureless, as no single change in an input pattern can change the output categorisation.

6.1.3. Distinction result with input pattern similar to CON_{Pass}

In this case the input pattern is most similar to the *Characteristic* pattern corresponding to the *Pass* output. Note that the actual categorisation produced by the trained neural network was a *Distinction*, notwithstanding the closer similarity of the pattern to that of a standard *Pass* student. We will extract rules here to explain the categorisation in terms of the difference from the relevant standard patterns. Note also that the *Characteristic* pattern CON_{Pass} is referred to as $COFF_{Dist}^{Pass}$ when it is being used as an *OFF* pattern for the *Distinction* output. The Causal Index graph is shown in Figure 11.

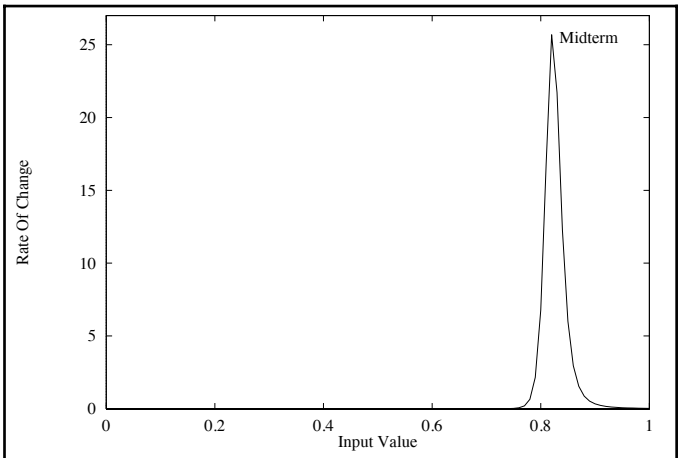

Figure 11: Causal Index of inputs for Mark Predictor network – $COFF_{Dist}^{Pass}$

The Causal Index graph shows that a change in a particular single variable can produce a *Distinction* result from an otherwise *Pass* result.

The most similar *Characteristic* pattern is shown in Table 2, as CON_{Cred}.

The rules produced are shown in Table 4.

Character. Pattern	Rule Set
$COFF_{Dist}^{Pass}$	Midterm ≥ 0.85

Table 4: Rules produced from the Causal Index graph for Mark Predictor network – $COFF_{Dist}^{Pass}$

This rule set is actually formed by modifying the rule set for the *Characteristic ON* pattern of the *Pass* output with the rule extracted for the case of a *Distinction* decision. Since there is only one term, it is completely replaced by the modified form.

What we have found from the extracted rule, is that there is a population of students who perform at a passing level in continuous assessment, but do particularly well in the *Midterm* examination and get a *Distinction* grade. We can readily accept that a good result in the *Midterm* will correlate with a good result in the final examination.

6.1.4. Distinction result with input pattern similar to CON_{Fail}

In this case the input pattern is most similar to the *Characteristic* pattern corresponding to the *Pass* output. The Causal Index graph for this case is shown in Figure 12.

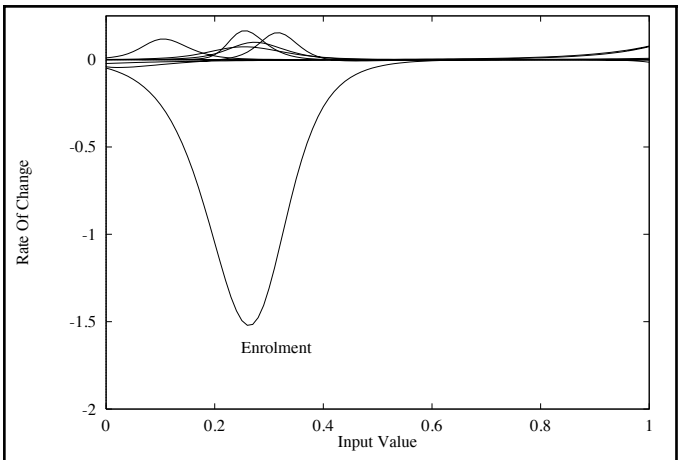

Figure 12: Causal Index of inputs for Mark Predictor network – $COFF_{Dist}^{Pass}$

As we can see, only one input, *Enrolment*, has any effect. This effect is too small to change a result which is overall consistent with a *Fail* to a *Distinction*. This is not truly suprising.

For completeness, the most similar *Characteristic* pattern is shown in Table 2, as CON_{Fail}.

In subsequent discussion, we will not display graphs with such insignificant effects.

7. Examples of Results

The network to be analysed is a three layer neural network with fourteen inputs, five hidden units and four output units. The network has been trained on a set of partial marks in a subject to predict the final mark that a student will receive [2, 3].

Example 1

Crs	0
Stg	0.5
Enr	0
Tutgp	0.25
lab2	0.4
TutAss	0.4
lab4	1
H1	0.8
H2	1.0
lab7	0.7
P1	1
F1	1
Mid	0.71
lab10	0

Table 5: Actual input pattern for a student – p0194588

p0194588
Network Output : Distinction
Most Similar Characteristic Input : Distinction
Important Inputs [Characteristic Values]
lab2 [0.7]
lab4 [1.0]
P1 [1.0]
Midterm [0.72]
Satisfied Rule Set
(lab2 ≥ 0.38) AND (lab4 ≥ 0.23) AND
(P1 ≥ 0.27) AND (Midterm ≥ 0.37)
Next Most Likely Output : Credit

This is a straightforward result, the most similar characteristic input pattern being that of the Distinction and the next most likely output being the *Credit*. For the mid-session quiz (*Midterm*), an input value of *0.72* is a good mark, the student performed well on key assignments, thus the result is consistent with experience.

Example 2

Crs	0
Stg	0.5
Enr	1
Tutgp	0.75
lab2	0.7
TutAss	0.7
lab4	0.7
H1	1
H2	0.8
lab7	0.4
P1	1
F1	0.3
Mid	0.48
lab10	0.5

Table 6: Actual input pattern for a student – p0185591

p0185591
Network Output : Credit
Most Similar Characteristic Input : Credit
Important Inputs [Characteristic Values]
Course [0.0]
Stage [0.5]
lab2 [0.85]
Midterm [0.59]
lab10 [0.5]
Satisfied Rule Set
(Course < 0.35) AND (Stage < 0.93) AND
(lab2 ≥ 0.39) AND (Midterm ≥ 0.12) AND
(lab10 ≥ 0.14)
Next Most Likely Output : Distinction

It is interesting to note that in this case the student actually failed the mid session exam, however the next most likely output predicted is a *Distinction*. This seems to be unusual, however the final grade was *71*, so it is likely to be correct. The student likely got a high mark in the final exam. Nevertheless, the mark was predicted by the network and explained even though the result was not obvious.

8. Summary and Conclusion

Our method of extracting meaning from neural networks to provide a useful explanation facility takes the following format:

- Characteristic inputs for each output pattern are calculated, each of these input patterns is then used as both characteristic ON and OFF inputs for the separate outputs of the network.
- The Causal relationship of the characteristic input with respect to the outputs is calculated.
- This relationship is used to determine the inputs of importance to the outputs of the network.
- Rule sets using these inputs are then generated.
- The input pattern is likened to the characteristic inputs, the characteristic input pattern showing the most similarity is selected and the important inputs and the satisfied rule set is presented.
- The next most likely output is presented.

This method only fails to produce the correct rule in 6% of cases for the training set of the network used. The separate specification of results for training versus test cases is not required since the measure of success is the replication of the network's result. That is, a rule is correct if it matches the conclusion rather than if the conclusion is correct.

By using our explanation method, many facets of the network's calculations can be determined. Firstly it becomes apparent which inputs are considered by the network for each output. The inputs considered for the *Distinction* class for example, are the inputs *lab2*, *lab4*, *P1* and *Midterm*. To test the accuracy of this, the set of all inputs classified as *Distinction*s in the training set had all other inputs set to zero. The resulting input patterns were input to the network, and in each case the *Distinction* classification was still chosen by the network. To examine the overall accuracy of the complete set of inputs considered important, all inputs not appearing in any of the set of inputs considered important to any of the outputs were set to zero in each pattern in the training set. The networks accuracy on the training set went down by 10% to 86%. Considering that over 40% of the input data has been discarded, this is a good result.

Our explanation facility provides explanations in the form of rules, which may be useful for expert system knowledge acquisition.

The rule set derived using this method is limited in application only by the selection of which rule is to be used. This is currently performed by likening the input pattern with the characteristic patterns.

The presentation of the next most likely output by the explanation facility is in this case only useful for the sake of interest of the user. Other applications of the next most likely output may include:

- Providing a 'safety net' in the case of incorrect classification, and
- Providing soft limiting boundaries – making decisions in 'grey areas' (such as classifying a mark of 64% as a Pass or a Credit) more flexible.

The accuracy of our method is affected by the size of the network's training set. *Characteristic* inputs will be most accurate using large training sets, which will allow more accurate statistical methods to be used in their generation. Nevertheless, we have achieved good results with a training set of only 84 patterns. Our method does not depend on the size or architecture of the network or on any unusual construction algorithm.

References

[1] Rumelhart, DE, Hinton, GE, Williams, RJ, "Learning internal representations by error propagation," in Rumelhart, DE, McClelland, *Parallel distributed processing*, Vol. 1, MIT Press, 1986.

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

[3] Turner, H and Gedeon, TD "Extracting Meaning from Neural Networks," *Proceedings 13th Int. Conf. on AI*, vol. 1, pp. 243-252, Avignon, 1993.

[4] Yoda, M, Baba, K and Enbutu, 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.

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

[6] Towell, GG and Shavlik, JW "The extraction of refined rules from knowledge-based neural networks," *Machine Learning*, August 1991.

[7] Bochereau, L and Bourgine, P "Expert systems made with neural networks," *International Joint Conference on Neural Networks*, vol. 2, pp. 579-582, January 1990.

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

[9] Sestito, S and Dillon, T "Automated knowledge acquisition of rules with continuously valued attributes," *Proc. 12th International Conference on Artificial Intelligence*, Avignon, 1992.