Semantic Extraction Using Neural Network Modelling and Sensitivity Analysis

TH Goh, Francis Wong
Institute of Systems Science
National University of Singapore
Heng Mui Keng Terrace, Kent Ridge
Singapore 0511
Email: ISSGTH@NUSVM.BITNET thgoh@iss.nus.sg

Abstract

Neural networks have often been faulted as black box systems that do not explain clearly the way it arrives at its conclusions. In this paper, we present three methods of using neural network modelling and sensitivity analysis to extract semantics from the historical data of a given system. First, neural network modelling and sensitivity analysis is used to determine the decision boundaries of the system. Several test cases including a simple credit rating system are described to illustrate the use and effectiveness of this method. Secondly, it is used for causal inferencing of the system. That is, determining which inputs has the largest effect on the output of the system. Typical problems like the decoder and parity are discussed. Lastly it is used to analyze the historical data for detection of exceptions or special input cases. In all three methods, the system being studied is first modelled using a neural network. Sensitivity analysis is then applied to the trained network to extract semantics learned by the network.

Introduction

Neural networks have often been faulted as black box systems that do not reveal the way its conclusions are reached. In this paper, we present three methods of using sensitivity analysis to extract semantics from systems that have been modelled using neural networks. Also in most real-world neural net applications, the greatest difficulties are in collecting good accurate data and selection of good input factors. The three methods we present attempts to solve some of these problems. Causal inferencing may be used to select good input factors or indicators and so that less useful indicators may be discarded. This will help reduce data collection and neural net training effort. Detection of special or exception inputs will be useful either to clean up the input data or to extract special cases for memory based or other types of system modelling. Finally, decision boundary detection may be used for better understanding of the system and hence may offer an explanation on the conclusions reached by the neural net model.

Since the wealth of neural networks lies in the connection weights within its network architecture, many approached the study of extraction of semantics through the analysis of the connection weights and the network architecture[1][2][3][4]. In this paper, we adopt a different approach. From historical data collected from the system under study, we first use a multi-layer perceptron neural network to model the system by training the neural network with the historical data. Having the neural network model, we then apply sensitivity analysis to this model to extract semantics and to analysis the system.

First, a method of determining decision boundaries of the system being studied is presented. Several test cases including a simple credit rating system are then studied to illustrate the use and effectiveness of this method. Secondly, it is used for causal inferencing of the system. That is, determining which inputs has the largest effect on the output of the system. Typical problems like the decoder and parity were discussed. Lastly it is used to analyze the historical data for detection of exceptions or special input cases.

I. Determining Decision Boundary

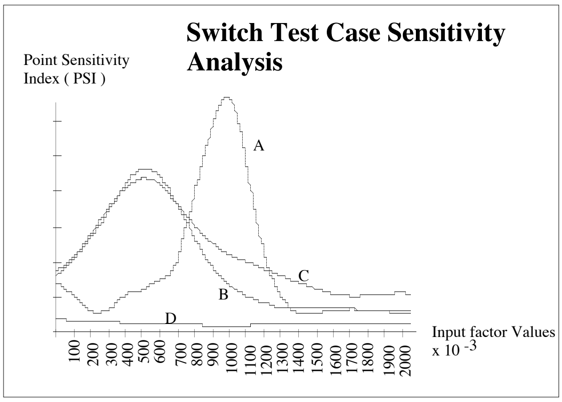
Many real life systems comprises numerous decision boundaries. A system that determines the credit rating of an individual may have many decision boundaries against the income of an individual. For instance individuals with incomes greater than $1000 may have good credit rating whereas one with less then $100, low credit rating.

In using sensitivity analysis for detecting decision boundary, we first calculate the point sensitivity index (PSI) for each input across the range of its value.

We calculate for each input , d the point sensitivity index, PSId

PSI_{dx}	for x = 0.01, 0.02 .... 3.0
$PSIdx = \Sigma \mid Oi(I_d{=}x) - Oi(I_d{=}x{+}\Delta) \mid$	for all i $Oi(I_d{=}x)$ is the output of the network for the input vector i but with the dth input replaced with the value x. $Oi(I_d{=}x{+}\Delta)$ is the output of the network for the input vector i but with the dth input replaced with the value x + Δ. Δ is a small value.

Having calculated the PSI for each input factor, the graph of the PSI for each input factor is plotted against the x value. Input factors that contains decision boundaries will show peaks in their PSI at those values where the decision boundaries are.



Switching Test Case

In this test case, there were 4 inputs **a,b,c,d** with a single output **f**.

The following rules were used to produce the output **f**;

If **a** < 1	THEN	**f = b**
	ELSE	**f = c**

d is random and had no effect on **f**

This test case is so termed because input **a** acts a switch allowing either **b** or **c** to pass through as the output **f**.
The plot of the point sensitivity index for input a shows a distinct peak at **a = 1**, indicating the decision boundary.

Credit Rating Test Case

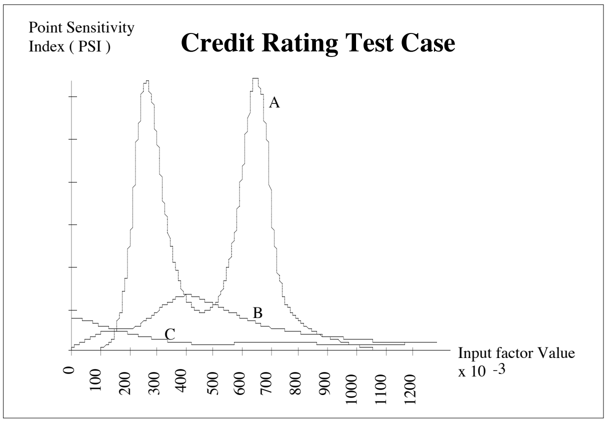
In this test case there were three inputs **A,B,C** and a single output **CR**.

The test data was generated using the following rules :-

IF	**A < 0.25**	THEN	**CR = 0**
IF	**A > 0.65**	THEN	**CR = 1**
IF	**0.25 < A < 0.65**	THEN	**CR = B.**

C is random and had no effect on CR.

This test case models a simplified credit rating system whereby if an individual's income (**A**) is less than $250 per month than his credit rating (**CR**) is poor. If his income is greater than $650 per month, than his credit rating is good. Otherwise, his credit rating depends on other factors, say his wife's income (**B**).



From the graph, two distinct peaks were observed for input factor **A** at values 0.25 and 0.65, indicating the possibility of decision boundaries existing at **A = 0..25** and **A = 0.65**.

II. Causal Inference

Previous work on using neural network for casual inference has centered on analyzing the weights of the network. In [1], the casual index is calculated using the sum of the products of the weights.
Comparison of the casual index thus calculated against the method presented here which we term sensitivity casual index shows that for most test cases, the results are similar. However for the case of the modified XOR, the causal index weighted heavily for one of the inputs which could not be inferred from the test data.

Sensitivity Causal Index (SCI)

For a network with i input vectors of dimension d with single output O,
for each input dimension or factor, the sensitivity causal index, S_d is calculated using

$S_d = \Sigma \mid \Delta S_{id} \mid$ for all i

$\Delta S_{id} = O(I_i) - O(I_i + \Delta_{id})$ I_i is the i^{th} input vector

$O(I_i)$ is the output of the network for the i^{th} input vector

$I_i + \Delta_{id}$ is the i^{th} input vector with a small value added to the d^{th} component.

The SCI was calculated for many test cases such as the parity 2, 3, 4 and the decoder 3, 4, 5. For the parity test cases, the SCI was almost equal in all cases. For the decoder test cases, for each output, the corresponding input returned significantly higher SCI..

For example the indices for the **decoder 5** problem are as shown on the table on the right. The SCI correctly shows the casual relationship between the inputs and the outputs.
The causal index shows similar results.

		OUTPUT				
		1	2	3	4	5
I	1	**14.9**	3.9	1.1	10.0	1.1
N	2	2.5	**13.4**	8.9	1.1	0.4
P	3	1.4	11.8	**15.0**	3.7	0.4
U	4	8.5	0.2	2.5	**11.9**	1.2
T	5	1.0	1.2	1.2	0.7	**6.4**

For the following **modified XOR** test case,

Inputs		Output
0	0	0
0	1	1
1	0	1
1	1	0
0.5	**0.5**	**1**

The causal index obtained were

Input	1	2
Causal Index	-4.3	**16.0**

Since the input data is symmetrical, the causal relationship for each input to the output should be the same. However the causal index thus computed showed incorrectly that input 2 has a higher causal relation to the output.

For the modified XOR test cast, the SCI obtained were

Input	1	2
SCI	0.3	0.3

indicating that both inputs are equally important as expected.

Switching Test Case

In this test case, there were 4 inputs **a,b,c,d** with a single output **f**.

The following rules were used to produce the output **f**;

If **a** < 1 THEN **f = b**
ELSE **f = a**

d is random and had no effect on **f**.

This test case is so termed because input **a** acts a switch allowing either **b** or **c** to pass through as the output **f**.

The SCI obtained were

Input	**a**	**b**	**c**	**d**
SCI	34	68	61	6

indicating correctly that input **d** does not influence the output **f**.

Hypothesis Testing

Suppose we hypotheses that the above rules apply to the studied system, we first divide the test data into 2 sets, the first set comprising only those with **a < 1** and the second set with **a >= 1**.

Then we compute the SCI separately for each set.

In this case, the SCI obtained were

Input		**a**	**b**	**c**	**d**
Set 1:	**a<1**	33.3	**115.8**	17.8	5.7
Set 2:	**a>=1**	31.5	19.6	**103.4**	5.8

Again the SCI correctly shows that

For A < 1, B has the greatest influence over F
For A >= 1, C has the greatest influence over F.

Note that this test may be difficult using the causal index.

III. Detection of Special Cases

Often with historical data, one may find exceptional (or erroneous) records or vectors in the historical data table. These exceptional or special records may be due to errors in the data collection or to special events that cannot be "generalized" with the rest of the data. For instance, data collected on trading in the stock market may be generalized with the exception of a few records collected during the beginning of the Gulf crisis. Once these special records may be identified, they may be "removed" and the remaining records may then be used to extract relationship rules between stock prices and the indicators monitored. Traditionally, euclidean distance based clustering techniques have been used to detect such special case and they work well in many instances. However in the equality test described below, such clustering techniques will not work well as the euclidean distance between the exception case and the rest is not large.

Vector Sensitivity Index (VSI)

To detect special cases in the data set, we compute the vector sensitivity index (VSI).

The VSI_i for ith vector I_i is computed thus

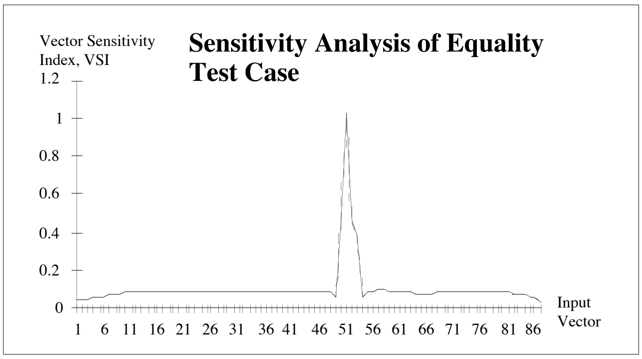
$VSI_i = O(I_i) - O(I_i + \Delta_i)$ $O(I_i)$ is the output of the network for the i^{th} input vector
$I_i + \Delta_i$ is the i^{th} input vector with a small value added to each component.

Equality Test Case for Detection of Exceptions

In this test case there was one input X with a single output Y.

where Y = 0.7 for X = 0.5
Y = A everywhere else.

This test case may be explained as follows, say the stock index (Y) for a particular counter may normally be forecasted by some leading composite index (X) with the exception of the one instance when X = 0.5. This exception may be due to factors not included in the composite index or due to erroneous data collected which cannot be easily verified. A table of 100 vectors observing the above rules was randomly generated. The exception case is then added to the table. A multi-layer perceptron neural network was trained on the table of data. After training, the input vectors are sorted and their vector sensitivity index computed. The plot of the VSI for each input vector is shown below.

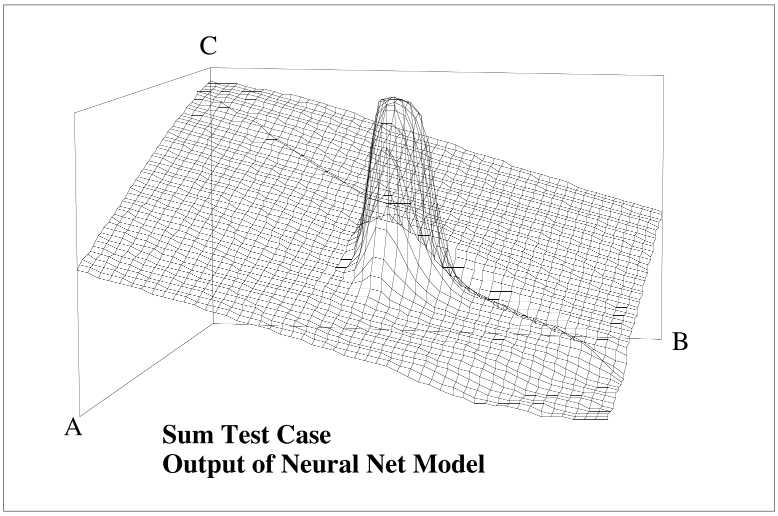


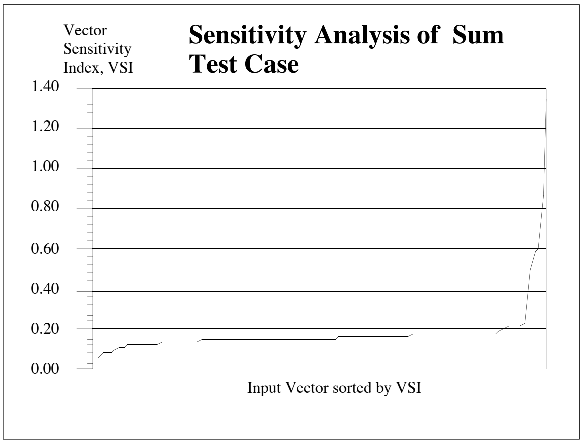
From the plot on the left, it can be seen that input vector 51 has the highest VSI value. This indicates the possibility of input vector 51 being a special or abnormal case that does not conform to the norm of the rest of the input vectors. By checking against the input vector table, as expected, input vector 51 corresponds to A = 0.5. Note that all those input vectors close to A = 0.5 also have high values of VSI.

Sum Test Case for Detection of Exceptions

In this test case there were two inputs A, B and a single output C.
100 random input vectors were generated using the rule

C = 1 where A = B = 0.25
C = A + B everywhere else





Again we created an exception or special case where A = B = 0.25. The data table created was trained on a 2-5-5-1 neural net. After successful learning, the VSI is computed for each input vector and sorted in ascending order. The plot is shown on the left. Again the highest VSI occurs at the input vector where A = B = 0.25 indicating the possibility of this being a special or exception case.

Concluding Remarks

We have outlined three methods where neural network modelling and sensitivity analysis may be applied. These method may be used to reduce the number of input indicators in a neural network application and hence reduce the data collection and neural net training efforts. They may also be used to clean up raw data or detect special instances in the input data for memory based or other types of system modelling. Lastly, decision boundary detection may be used for better understanding of the system and hence may offer an explanation on the conclusions reached by the neural net model. Presently we are applying these methods to real world stock price data and economic data. There are other ways that this combination of neural network modelling and sensitivity analysis may be used. We are working on using this technique as a measure of the confidence level of our neural networks predictions.

References

1. Kenji Baba, Ichiro Enbutu and Mikio Yoda, Explicit Representation of Knowledge Acquired From Plant Historical Data Using Neural Network, IJCNN 90, III - 155 to 160, 1990.

2. Bochereau L. and Boutgine P., Extraction of Sematic Features and Logical Rules from a Multilayer Neural Network, IJCNN 90 Wash DC, Vol. 2, II-579 to 582, 1990.

3. Sabrina Sestito and Tharam Dillon, The Use of Sub-Symbolic Methods for the Automation of Knowledge Acquisition for Expert Systems, Avigon 91 : Expert Systems and their Applications, France, 1991.

4. Kazumi Saito and Ryohei Nakano, Rule Extraction from Facts and Neural Networks, INNC 90, pg 379 - 382, 1990.

5. D.E. Rumelhart, JL McClelland and the PDP Research Group, Parallel Distributed Processing, Vol. 1, MIT press, 1986.