GLOBAL JOURNAL OF PURE AND APPLIED SCIENCES VOL. 17, NO.2 2011: 159-164
COPYRIGHT© BACHODU SCIENSE CO. LTD PRINTED IN NIGERIA ISSN 1118-0579
www.globaljournalseries.com, Email: info@globaljournalseries.com

PREDICTING MORTALITY IN HEPATITIS-C PATIENTS USING AN ARTIFICIAL NEURAL NETWORK

OLUMIDE OWOLABI AND BARILEÉ B. BARIDAM

(Received 1 August 2008; Revision Accepted 20 July 2010)

ABSTRACT

We have developed an artificial neural network that is capable of predicting whether a patient suffering from the hepatitis-C virus is likely to live or die. With test data, the system achieved 70% accuracy in determining when a patient would live and 60% accurate in determining when a patient would die. It is hoped that with further work, the accuracy of the system will be considerably improved.

KEYWORDS: Artificial neural networks, disease diagnosis, prognosis, hepatitis-C, machine learning.

1. INTRODUCTION

A neural network is an information processing paradigm that is inspired by the way the biological nervous system (i.e., the brain) processes information (Siganos, 1994). It can also be defined as a massively parallel distributed processor that has a natural propensity for storing experimental knowledge and making it available for use (Haykin, 1994). It resembles the brain in two respects: Knowledge is acquired by the network through the learning process; and interneuronal connection strengths known as synaptic weights are used to store knowledge.

Neural networks are intuitively appealing because they are based on a low-level model of the biological nervous systems. The power of the brain comes from the number of these neurons and the multiple connections that exists between them. An artificial neural network follows the same principle. The main aspect of this paradigm is the processing system. It consists of a large number of highly interconnected processing units called neurons working in unison to solve specific problems. An artificial neural network is such that it is configured for just a specific application and learns from given examples in the application domain. Learning in biological systems involves adjustment to the synaptic connections between neurons; it is the same for neural networks.

An artificial neural network has the following properties: (Patterson, 1996)

- It can receive a number of inputs and each input comes via a connection that has a strength which corresponds to the synaptic efficacy in a biological neuron. Since each neuron has a single threshold value, the activation of a neuron can be gotten by subtracting the total threshold from the weighted sum of inputs.
- An activation signal is passed through an activation or transfer function to get the output of the neuron.

Neural networks have the remarkable ability to derive meaning from complicated and imprecise data; thus they can be used to extract patterns and detect trends that are too complex to be noticed by humans or other computing techniques. A trained neural network can be thought of as an “expert” in the field it is given to analyze. It can be used to provide projections given new situations of interests. Other advantages include: the ability to learn how to do tasks based on the data given for training or initial experience; the ability to create its own organization or representation of the information it receives during learning time; computations can be carried out in parallel; and it has the ability to operate effectively on incomplete data.

Neural networks are applicable in virtually every situation in which a relationship between the predictor variables (independents, inputs) and predicted variables (dependents, outputs) exists. These networks have been built for tasks such as prediction, pattern recognition, classification, optimization, data association, data conceptualization, and data filtering. Although these networks have been found in such a wide range of applications, they suffer the drawbacks common to other forms of expert systems. These include the inability to capture deep knowledge as well as difficulty in clearly explaining their actions (Liebowitz, 1997).

The ability of neural networks to associate a given set of inputs with some particular output pattern could be exploited to predict the outcome of an illness given a set of symptoms. The aim of this project is to develop a neural network that accepts the symptoms of the Hepatitis-C virus in patients as input variables. The network then determines, based on past observations, whether or not a patient will survive the illness. The data for the work were collected from Redcol Clinic, Port Harcourt. The knowledgebase used was obtained from interviews with doctors in the clinic and from the literature (Novartis, 2003; www.nlm.nih.gov/medlineplus/ency/article/).

2. NEURAL NETWORK ARCHITECTURE

The basic unit of a neural network is the artificial neuron depicted in Fig. 1.

Olumide Owolabi, Computer Centre, University of Abuja, PMB 117, Abuja, Nigeria.
Barileé B. Baridam, Department of Computer Science, University of Port Harcourt, Port Harcourt, Nigeria.

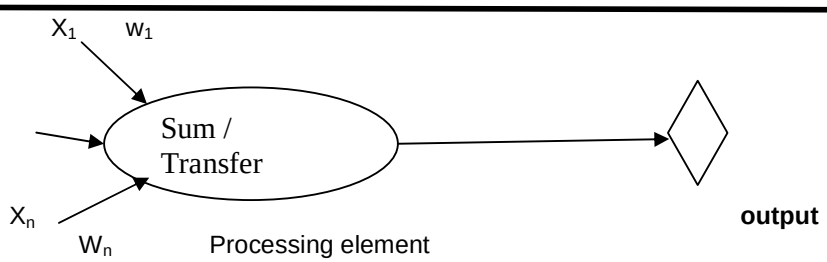


Fig. 1: An artificial neuron

In the figure, $X_1, X_2,\ldots X_n$ are the inputs to the network, while $W_1, W_2,\ldots W_n$ are the respective connection weights. Each input is multiplied by the appropriate connection weight, the products are summed and fed through a transfer function which generates an output (Patterson, 1996). These neurons are connected in a distinct layer topology to get a network. Some neurons are in the input layer, some others in the middle, that is, the hidden layers, while the rest are in the output layer. A neuron has two modes of operation, the training and the use mode. In the training mode, the neuron is taught to fire (or not) for a particular input pattern while in the use mode, when a taught pattern is detected at the input, its associated output becomes the current output.

A basic network has a feed-forward structure which means that signals flow from the input, forward through hidden units to the output units. A single layer feed-forward network usually consists of inputs, a single layer, the output, and maybe, a bias. Most multilayered networks consist of three or four layers made up of one input layer and one output layer with one or more hidden layers depending on the problem. (Fröhlich, 1997). The input serves to introduce the values of input variables. The hidden and output layers are all connected to the units of the preceding layer. Generally when artificial neural networks are executed, the input variable are placed in the input neurons and the hidden and output neuron are progressively executed. Each of them calculates its own activation value by taking the weighted sum of the output units in the preceding layer. The activation value is then passed through an activation function to produce the output of the neuron. The outputs of the output layer acts as the output of the entire network.

3. LEARNING IN NEURAL NETWORKS

Neural networks have to be trained to associate a given outcome with a given set of inputs. While learning different inputs, the weight values are changed dynamically until their values are balanced so each input will lead to the desired output (Haykin, 1994). Usually the training of a network leads to a matrix that holds the weight values between the neurons and once it has been trained correctly, it can find a desired output using these values. Since there are certain errors in the learning process, the generated output is an approximation of the perfect output.

About the commonest learning algorithm employed in multilayered networks is the back propagation algorithm, which is a special form of the Delta learning rule (Fröhlich, 1997). The algorithm uses the computed output error to change the weight values in the backward direction like the name implies. The net error is first gotten by using the phases in the forward propagation algorithm. During the forward propagation, the output of each neuron in the hidden layer is passed through the sigmoid activation function:

$$f(x) = \frac{1}{1+\exp(-x)}$$

A neuron fires, i.e., contributes to the output of the network, only if the value of the function exceeds the threshold value for the neuron.

Activation functions for the hidden units are needed to introduce non-linearity into the networks. Without nonlinearity, hidden units would not make nets more powerful than just plain perceptrons (which do not have any hidden units, just input and output units). The reason is that a composition of linear functions is again a linear function. However, it is the non-linearity (i.e., the capability to represent nonlinear functions) that makes multi-layer networks so powerful. The sigmoid function is the most commonly used activation function in neural networks (Fausett, 1994). The sigmoid output varies continuously but not linearly as the input changes and has been found to bear a great resemblance to the behavior of real neurons (Alexander and Morton, 1995; McCullough and Nelder, 1989).

The neural network element computes a linear combination of its input signals, and applies the bounded sigmoid function to the result; this ensures that the output is kept within desired range, that is, 0 - 1. Inputs entering a neuron not only get multiplied by weights, they also get multiplied by the neurons characteristic equation, or transfer function. The sigmoid function is a typical neuronal non-linear transfer function that helps make outputs reachable. The non-linearity is significant for a further reason. If the transfer function were linear, each of the neuronal inputs would get multiplied by the same proportion during training. This could cause the entire system to "drift" during training runs. That is, the system may lose outputs it has already tracked while attempting to track new outputs. A non-linearity in the system helps to isolate specific input pathways. (Anderson, 1995; Nelson, 1990). The behavior of the sigmoid function is shown in Figure 2.

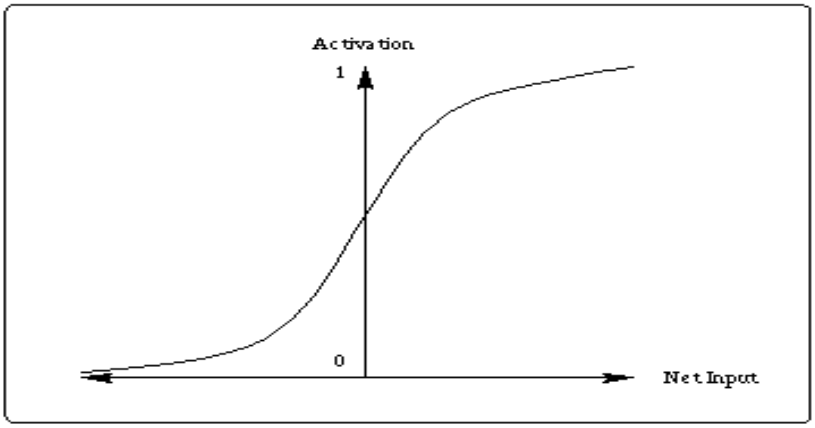


Figure 2: Sigmoid Activation Function

The training algorithm works as follows:
Step 1: Perform the forward propagation phase for an input pattern and calculate the output error.
Step 2: Change all the weight values in the weight matrix using the formula Weight (old) + value of change in weights (cw),

where

cw = learning rate * output error * output (neuron i) * output(neuron i + 1)

* [1 – output (neuron i + 1)]

Step 3: Go to step 1
Step 4: The algorithm is repeated until the maximum number of training sessions have been executed.

The first input pattern is propagated through the network using randomly selected initial weights. The same procedure is carried out for the next input pattern but now with the changed weight values. After the feed-forward and back ward propagation of the second to the last input patterns, one learning step is completed and the grand error is calculated by summing the square of the output errors of all the patterns. This procedure is repeated until the error gets to zero or an approximation of zero.

The training process can be seen as an optimization problem, where we wish to minimize the mean square error of the entire set of training data. Over the training step, the network converges to an optimal set of weights, which gives a network that can handle any arbitrary pattern that belongs to the same class as the training set. So, the training normally starts with a set of arbitrary values in the range -1 to +1 and can even be set using a random number generator. The back-propagation training step ensures that each of the weights converge to its optimal value for the network (McClelland and Rumelhart, 1988).

4. CHARACTERIZING THE HEPATITIS-C VIRUS INFECTION

Hepatitis-C infection is a disorder in which the virus and its products result in the inflammation of the liver cells resulting in their injury or destruction (Novartis, 2003). Damage to the liver can impair some vital body processes. Hepatitis C infection varies in severity from self-limited condition with total recovery to life threatening cases. It can take the acute form which may last up to two months or the chronic form which persists for prolonged periods.

The methods for diagnosing the presence of hepatitis C include:

1. Blood test - in people suspected of having or carrying the virus, certain substances in their blood will be measured. The substances include:

a. Bilirubin- it is one of the most important factors indicative of hepatitis. It is a yellow-red pigment normally metabolized in the liver and excreted in the urine. In people with hepatitis, the liver cannot metabolize bilirubin leading to its high level in the blood which leads to jaundice.

b. Liver enzymes- these enzymes such as aspartate [AST] and alanine [ALT] are released when the liver is damaged. Measurements of these enzymes in the blood is indicative of hepatitis C in a patient.

2. Radioimmunoassay- special blood tests known as Radioimmunoassay are performed which identifies particular antibodies which are molecules in the immune system that attack specific antigens. The assay for individual hepatitis virus may differ.

3. Liver biopsies- a liver biopsy may be performed for acute viral hepatitis caught in a late stage or for severe cases of chronic hepatitis. Normally a biopsy helps to determine the extent of damage to the liver.

Some of the symptoms of the virus, which may develop about a month after a person is infected, include itchy skin, fatigue, pain in joints, jaundice and cirrhosis. Some of the more marked symptoms include:

- A large swollen abdomen known as ascites with stomach and intestinal bleeding;
- Mental confusion, coma and a peculiar hand flapping tremor called asterixis.
- Cirrhosis which occurs when liver cells are destroyed between the portal tract and central veins of the liver which leads to a buildup of a layer of scar tissue over the liver

5. NETWORK DESIGN

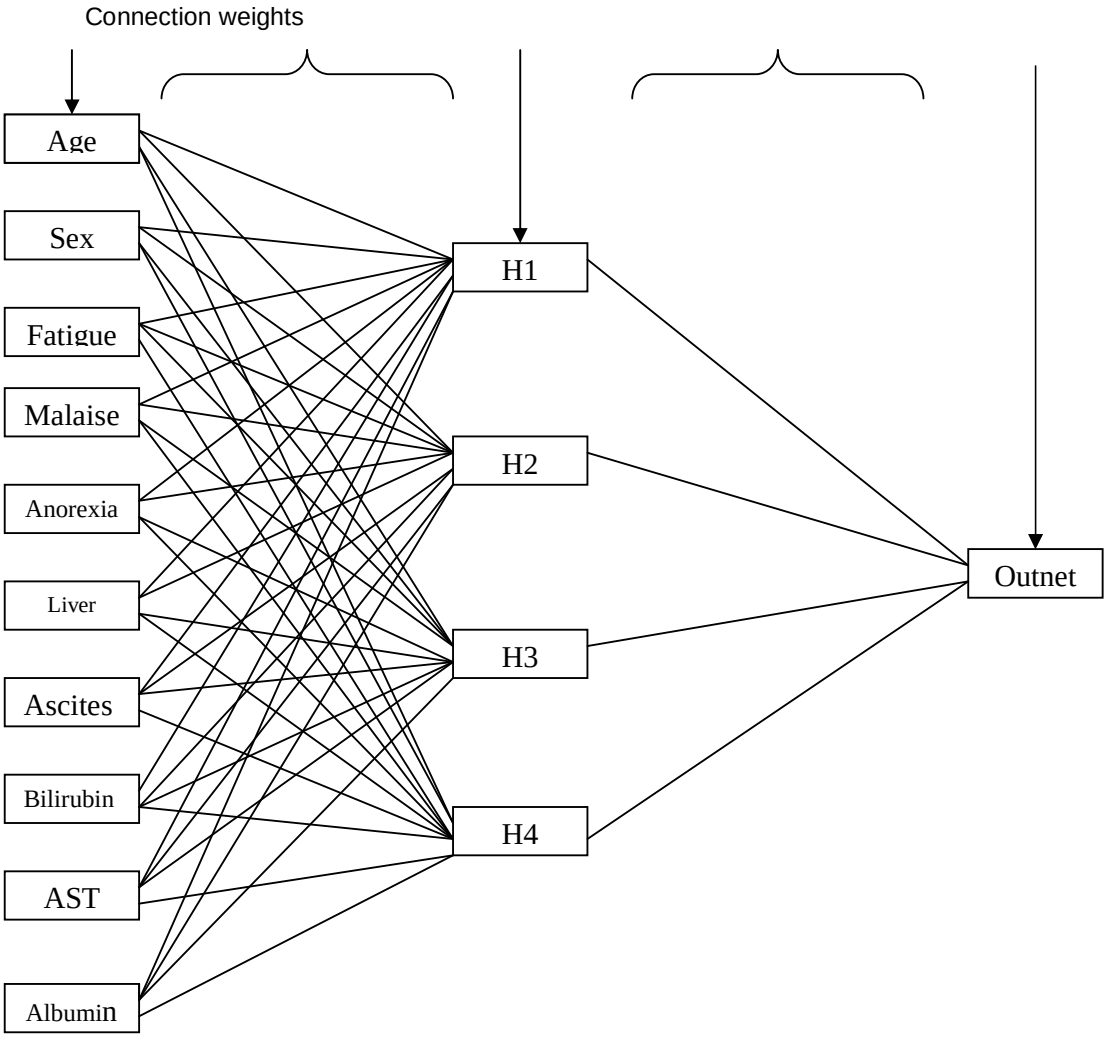
The network architecture used in this work is the multilayered type with three layers. The layers are the input, hidden and output layers. The network consists of ten input neurons, four hidden neurons and one output neuron. The architecture is shown in 3. The nomenclature used in the diagram is as follows:

I1 – I10: the first input neuron to the last input neuron;
H1 – H4: the first hidden neuron to the last hidden neuron;
Outnet: the output neuron;
W_{xi} – W_{yj}: weight connections from neuron i in layer x to neuron j in layer y.

The data for this work was gotten from Redcol Clinic in Port Harcourt. The major symptoms of the hepatitis-C virus are about twenty in number but for the convenience of the network, only ten major symptoms were selected for the network to learn sufficiently.

The attributes used are the following:
a. Age: 10-80
b. Sex: Male or Female
c. Fatigue: No or Yes
d. Malaise: No or Yes
e. Anorexia: No or Yes
f. Firm liver: No or Yes
g. Ascites: No or Yes
h. Bilirubin: 0.3-1.0 mg/dl (normal adult levels)
i. AST: 10-34 IU/litre (normal adult levels)
j. Albumin: 3.5-5.5 gm/dl (normal adult levels)
k. Output: Live or Die

Out of the ten input variables selected, six are nominal while four are continuous. The data was subjected to an attribute transformation process to fit the data for neural network processing. Artificial neural networks process numeric data at a fairly limited range (Bishop, 1995). Continuous variables, in particular, might have to be normalized by scaling them to an appropriate range for the network. The non-numeric data like sex = {male or female} and fatigue = {yes or no} could be represented as 1 or 0. An example of a case that has been transformed for the network to use is shown in Table 1.



3: Network Architecture.

Table 1: Transformed input data.

Variable	Raw value	Transformed value
Age	27	0.27
Sex	Male	1
Fatigue	Yes	1
Malaise	No	0
Anorexia	Yes	1
Liver firm	Yes	1
Ascites	Yes	1
Bilirubin	0.90	0.90
AST	63	0.63
Albumin	4.7	0.47

6. NETWORK ALGORITHM

The network processing algorithm used in this work is as follows:

Step 1: Initialize all the weights needed in the network
Step 2: Read all the input variables from all the cases from a file and target output from another file
Step 3: Train the network using the input variable and target output read from the file to get network error
Step 3.1: Run the values of the input variables and weights to get the network output.
Step 3.1.1: Process the hidden layer to get the output of the neurons in the hidden layer.
Step 3.1.2: Process the output layer to get the output of the output layer.
Step 4: Alter the weights in the network by using the error gotten from the previous case.
Step 5: Go to Step 2 and repeat until the 150^{th} case.
Step 6: Calculate the grand error by summing the square of the errors in each case.
Step 7: Repeat all the steps until the grand error is less than or equal to the set limit error or the maximum number of training sessions is reached
Step 8: Save the last set of altered weights used in the training.
Step 9: Test the network by supplying it with input values using the last saved weights.
The cases that were available for the network were limited based on the scarcity of data on hepatitis-c virus.

The network was implemented as a C++ program with several modules, the most important of which are: Initialize, Train, ProcessHiddenLayer, ProcessOutputLayer, SigmoidHidden, SigmoidOutput, ThresholdOutput, ThresholdHidden and AlterWeights. The function Initialize sets up the initial weight matrix, while ProcessHiddenLayer and SigmoidHidden compute the outputs of the neurons in the hidden layer. ThresholdHidden then applies the threshold function to the hidden layer neurons to determine which to fire. The outputs of this function go into the function ProcessOutputLayer, which together with SigmoidOutput computes the outputs of the output layer. ThresholdOutput decides which of the output neurons will contribute to the final output of the network. The function Train is used in the training of the network; it repeatedly calls the function AlterWeights to adjust the weight matrix using the back-propagation method until the training is completed.

The network was trained with a learning rate of 2.0 and a learning error of 0.05. The weights used for the training were random numbers ranging from -1.0 to +1.0. The weights indicate the connection strengths and are adjusted to their appropriate values during training. The maximum number of training sessions was set at 15 assuming the network does not converge to the learning error of 0.05. The number of cases used in training was 150 in number with 72 of the cases representing patients that lived and the remaining 78 cases representing cases of patients that died. This is about the usual training size for neural networks. The minimum network error gotten from this network was 6.95432 after the maximum 15 sessions. The last set of weights used in training was saved for use in testing the network.

7. TEST RESULTS

The total number of cases used to test the network was 20 consisting of 10 “lived” and 10 “died” cases. These cases were selected at random from the pool of 150 cases. It was observed that out of the ten cases whose output was “live” (1), seven of them fell into the range of 0.46 - 0.54 with the rest less than 0.40. It was also observed that out of the ten cases whose output was “die” (0), six of them fell into the range of 0.17 – 0.32 while the rest fell outside this range. The results are shown in Table 2. From these results, it can be concluded that the network is 70% accurate in predicting cases of patients that would survive and 60% accurate in predicting cases of patients that would die.

The interpretation of the network output is essentially a discretization process in which network outputs ranging from 0.0 to 1.0 are converted to a binary value ‘Yes’ or ‘No’. As usual with such process, values from 0.4 up are converted to 1 while the rest are converted to 0. This is to accommodate a ±10% error around the 0.5 mark. The network has learnt to accept these value ranges during the training step.

Table 2: Test results for the neural network.

CASE NUMBER	RESULT	KNOWN OUTPUT
1	O.47834	1
8	0.18134	1
4	0.47421	1
39	0.47159	1
25	0.54262	1
12	0.46285	1
42	0.46677	1
14	0.32865	1
45	0.47628	1
79	0.22469	1
140	0.17466	0
146	0.39041	0
23	0.32136	0
130	0.62506	0
136	0.26811	0
96	0.32242	0
150	0.54877	0
143	0.27543	0
125	0.31143	0
7	0.40243	0

8. CONCLUSION

Neural networks are powerful classification tools when applied to multiple variables extracted from individual cases like in this application. In this application, they help to predict the outcome of hepatitis-C infections. The network described in this paper has achieved a success rate of between 60 and 70% in predicting the mortality of hepatitis-C patients. The success rate achieved using the limited input data available and the few number of test cases suggest that it would be worthwhile to carry out more extensive testing on the network. In order to make this neural network more realistic and accurate in its classification, we hope to experiment on the network architecture with a view to either increasing the number of hidden layers or increasing or reducing the number of units in the hidden layer. The number of training cases used will also be substantially increased and more extensive testing carried out.

REFERENCES

Aleksander, I. and H. Morton, 1995. An Introduction to Neural Computing (Second Edition), Thomson Computer Press.

Anderson, J. A., 1995. An Introduction to Neural Networks. Cambridge, MA/Bradford Books.

Bishop, C., 1995. Neural Networks for Pattern Recognition. Oxford University Press.

Fausett, L., 1994. Fundamentals of Neural Networks: Architectures, Algorithms, and Applications. Prentice-Hall, Inc.

Fröhlich, J., 1997. Neural networks with java. http://pages.cpsc.ucalgary.ca/~carman/533/progress/progress_report.doc

Haykin, S., 1994. Neural Networks: A Comprehensive Foundation.: Macmillan Publishing, New York.

Liebowitz, J., 1997. Handbook of Applied Expert Systems.: CRC Press, Boca Raton, Florida.

McClelland, J. L. and D. E. Rumelhart, (Eds.). 1988. Explorations in parallel distributed processing: A handbook of models, programs, and exercises. Cambridge, MA: MIT Press.

McCullough, P. and J. A. Nelder, 1989. Generalized Linear Models, 2nd ed., London: Chapman & Hall.

Medlineplus. www.nlm.nih.gov/medlineplus/ency/article/.

Morgan, N., ed. 1990. Artificial Neural Networks : Electronic Implementations. Los Alamitos, CA: IEEE Computer Society Press.

Novartis, 2003. Foundation for gerontology. ttp://directory.tiscali.it/Health/senior_Health

Patterson, D, 1996. Artificial Neural Networks. Prentice Hall, Singapore:

Siganos, Dimitrios, 1994. Neural Networks. www.doc.ic.ac.uk/~nd/surprise96/journal/vol4/cs11/report.htm