International Journal of Technology and Science, ISSN (Online) 2350-1111, (Print) 2350-1103 Volume 3, Issue 3, 2016, pp. 9-12

R Library For Neural Networks

V M Aparanji[1], Uday V Wali[2], and R Aparna[3]
[1] Dept of E&C, SIT, Tumakuru, India
[2] Dept of E&CE, KLE Society's Dr.MSS CET, Belagavi, India
[3] Dept of ISE, Siddaganga Institute of Technology, Tumakuru, India
Email: [1]vmasit@rediffmail.com, [2] udaywali@gmail.com, [3] raparna@sit.ac.in

Abstract— **With growth in applications using neural networks, new types of network configurations and learning mechanisms are evolving. New types of Artificial Neural Network (ANN) models are being developed that implement new ANN models. In this paper, we have presented a new library of ANNs that includes some of such newly proposed models along with the traditional ANNs. This proposed library of ANNs can be used to implement, test and use new types of ANNs. The library contains three major types of ANNs, viz., Back Propagation Network (BPN), Adaptive Resonance Theory (ART) and Self Organizing Map (SOM). Multiple implementations for each of these types of ANNs based on models suggested by various authors are available in the library. A novel ART2 type of network model developed by us is available as a part of this library. The paper discusses implementation and usage details of this set of ANNs.**

Index Terms— **Adaptive Resonance Theory (ART), ART1, ART2, Artificial Neural Networks (ANNs), Back Propagation Network (BPN), Self Organizing Map (SOM).**

I. Introduction

This paper discusses the implementation of neural networks in R programming language that can be used as tools for neural network research or application development.

R is used for statistical data modelling [1]. It is similar to other application level languages, like the one defined by MATLAB[2]. R has extensive computer graphics capabilities which helps to visualize the data and computed results. Standard libraries of R provide support for most of the scientific, engineering and web applications, e.g., Mathjax, JSON, jQuery, Datatables, Google web APIs, Qt, etc. We have selected R because of these capabilities. Note that R Language is available under GNU Affero General Public License.

This paper completely skips descriptions of various types of ANNs that are widely available in literature [3]. This paper explains new ART2 - Short Term Memory (STM) based model developed by us. We have included an overview of the new ART2 structure. However, full details of this type of ANN are being published separately.

II. Artificial neural network (ANN)

ANN consists of programming constructs called neurons, which perform the complex data transformation functions. ANNs acquire knowledge from the environment through a learning process [3]. Synaptic weights store the acquired knowledge. Neural networks are widely used in applications like data classification, pattern recognition, character recognition, etc. Newer models of ANN with improved cognitive and predictive performance. These models are finding applications hither to considered NP-hard, e.g., natural language processing and scene recognition. As new types of ANNs will not be currently available on existing libraries, we will find it difficult to test and improve suggested ANN models. Therefore, it is necessary to build ANN libraries ourselves, which we can test and enhance. We have developed one such library that includes traditional ANNs as well as new proposed models of ANNs. Over period of time, the library will be made available to public under GPL or similar license.

III. Ann library

Several models of ANNs are implemented in this library. Different types of ANNs can be largely grouped into the following main categories:

- Back Propagation Network (BPN)
- Adaptive Resonance Theory (ART)
- Self Organizing Map (SOM)

Some of the models available in the proposed library are discussed here.

A) Back Propagation Network (BPN)

BPN consists of one input layer, one or more hidden layers and one output layer. BPN makes use of supervised learning.[4]. Expected output for a given set on input values is used to train a BPN. Once trained, a BPN can predict the expected output with fair amount of reliability [7]. Therefore, BPN implementation is divided into two parts, first one is *Learning BPN* in which network is trained for the known set of input and output samples until the error becomes acceptably low. Second one is *Trained BPN* in which weights obtained from the *Learning BPN* is used. Random inputs and random weights are used in *Learning BPN*, whereas only random inputs are used in *Trained BPN*.

The following steps are used to train BPN

- Output is calculated for a given set of input through the forward path which consists of a layer of multipliers (axons) and non-linear sigmoid limiting function.
- Difference between the actual output and the target output (Error) is back propagated to adjust the weights until the difference becomes acceptably low.

Once the network is trained, following step is required to estimate the output.

- The weights obtained from the training are used to get an output of ANN with any input data set [12].

User has to specify i, j, k, n, η and α, where,

International Journal of Technology and Science, ISSN (Online) 2350-1111, (Print) 2350-1103 Volume 3, Issue 3, 2016, pp. 9-12

i implies number of input neurons (out of which one input i.e. LSB is the bias input, which is always set to 1).

j implies number of hidden neurons.

k implies number of output neurons.

s implies number of input samples.

n implies number of iterations.

η is the learning parameter.

α is momentum parameter.

As an illustration, the following input parameters are considered to implement an XOR function to train the BPN,

$i = 3,\ j = 12,\ k = 1,\ \eta = 0.005,\ \alpha = 1,\ s = 400,$

$n = 1000$ (each input sample is trained for n/s times.)

Inputs are set to a range of -4 : 4 except the bias input which is always equal to 1.

Figure 1 shows the results obtained during the training of $3-12-1$ BPN (3 input neurons, 12 hidden neurons and 1 output neuron). The network is trained for 1000 iterations. There is a difference between the target output and the actual output. The error starts converging after 600 iterations.

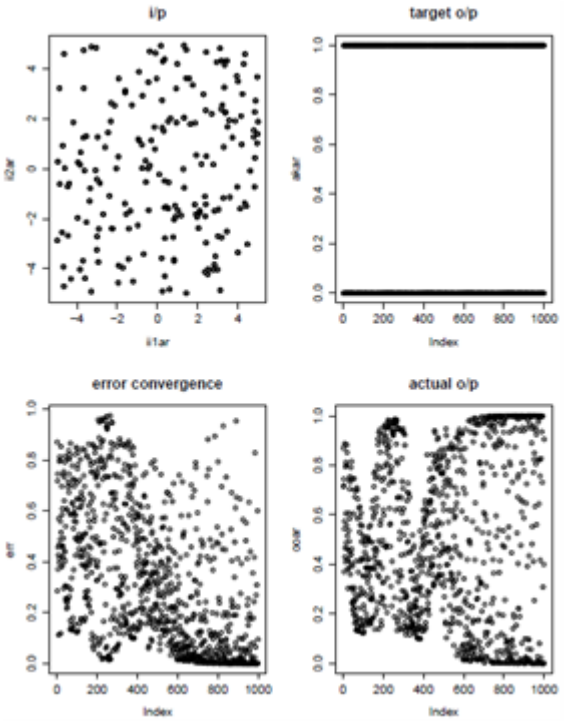

Figure 1 : Error convergence of $3-12-1$ *Learning BPN*

Same input parameters are considered for *Trained BPN* except number of samples s and number of iterations n (s and n are set to 40 and 20 respectively). The expression used to threshold the output is as follows,

$$f(o) = \begin{cases} 0 & if\ o \leq 0.2 \\ 1 & if\ o \geq 0.6 \\ o & else \end{cases}$$

The weights obtained from the *Learning BPN* are used in *Trained BPN*. Figure 2 shows the results for $3-12-1$ *Trained BPN*. The actual output meets the expected output for most of the inputs. Some errors in predicted output are characteristic of any ANN. In this example, success rate is about 95%. It is important to know that in most other cases success rate will be much lower.

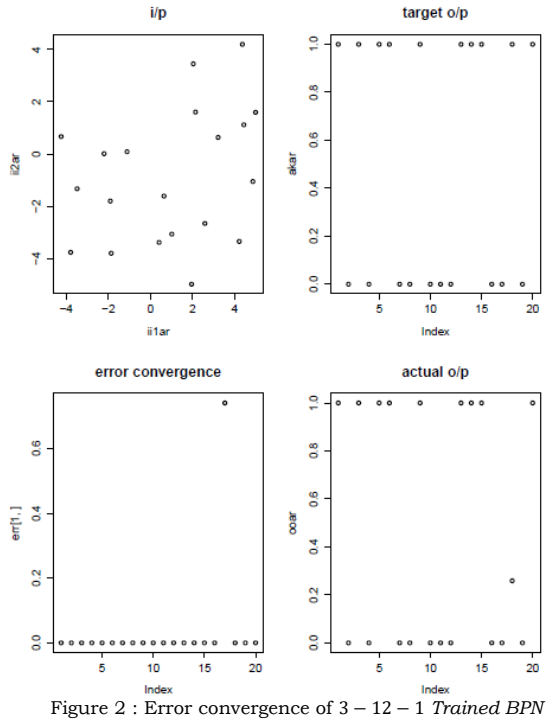

Figure 2 : Error convergence of $3-12-1$ *Trained BPN*

One of the problems faced in ANNs is the issue of stability. If the learning rate is high, the results tend to overshoot or undershoot, resulting in unusable results. On the other side, if the learning rate is low, the network may not stabilize and keep monotonically approach final values even for a large training set. Another problem with BPN is that the network configuration is static in the sense that the number of outputs and inputs is fixed. Real life learning will require changes to the network structure when new situations arise that do not match training set. Therefore, ANNs that attempt to simulate human like response cannot be really implemented using BPN. This is often called the Plasticity Stability Dilemma. Networks built using ART can handle such situations much better.

B) . Adaptive Resonance Theory (ART)

ART uses unsupervised learning. It overcomes the Plasticity Stability Dilemma by dynamically adding end nodes to existing network. ART networks are able to grow additional neurons if a new input cannot be categorized appropriately with the existing neurons [5]. A vigilance parameter ρ determines the tolerance of this matching process [6]. Several variations of ART networks exist.

A. ART1

ART1 networks accept binary inputs and output values are also binary. We have implemented ART1 as a part of the library. Users can specify number of input neurons i, vigilance parameter ρ which ranges between 0 : 1, maximum number of output clusters j to be recognized and the number of iterations n.

International Journal of Technology and Science, ISSN (Online) 2350-1111, (Print) 2350-1103 Volume 3, Issue 3, 2016, pp. 9-12

4-bit binary numbers are considered as an inputs for an illustration with the following input parameters,

i = 4, j = 1, n = 15.

Figure 3 shows the output clusters generated with different vigilance parameter (ρ). Here, we start with a single node corresponding to the first input. Every time a new pattern, which cannot be recognized by the output clusters arrives as an input to the network, a new node is added by the network at output node. It can be noticed that the number of output clusters increases as the vigilance parameter increases. Group of circles of same color represents a single output cluster.

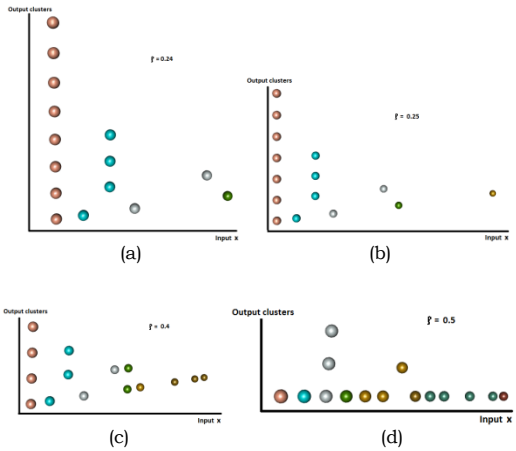

Figure 3: Output of ART1, (a) ρ = 0.24 (b) ρ = 0.25 (c) ρ = 0.4 , (d) ρ = 0.5

B. ART2

ART2 networks accept real inputs and generates real outputs. ART2 is implemented for i number of inputs and j number of output clusters. User needs to specify the number of inputs i, number of outputs j, the number of iterations n, values of vigilance parameter ρ and learning parameter η which ranges from 0 : 1.

As an illustration the following input parameters are considered,

$i = 2, j = 1, \rho = 0.9, \eta = 0.6, n = 50$

Input values are real numbers in the range -1:1.

The output of ART2 is shown in Figure 4. ART2 network is implemented to cluster the inputs based on the quadrants of the input. Therefore, though the output cluster was initialized to one, three more clusters are created Group of circles of same color represents a single output cluster.

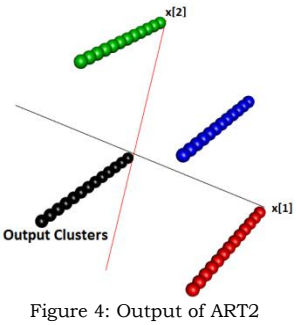

Figure 4: Output of ART2

C. ART2: A new ART2 Model

We have proposed and implemented a completely new structure similar to ART2. Figure 5 shows structure of new ART2. The diagram is shown for a single output cluster. There are m units of each of these types to represent m clusters, where m is the maximum number of clusters generated at the output. When an input appears at the input neurons, the recognition nodes resonate if the input matches the corresponding Short Term Memory (STM). Other nodes will be out of resonance. Resonance is achieved if and only if all the paths from input to recognition layer in a cluster are in resonance. If none of the nodes are in resonance, a new node is added. The process of adding a new cluster starts when none of the nodes in the recognition layer is in resonance. Adding a new cluster involves identifying a set of input multipliers that set the new cluster into resonance. As other clusters are not resonating, the new cluster will represent the input better than all other existing clusters. The process continues to add new clusters depending on how individual clusters spread their areas of resonance. This can be controlled by using statistical parameters from the input stream. Some of the clusters may resonate very sharply at a given set of input while the others may resonate over a range of input values. For example, in Figure. 6, selecting a different threshold (t) results in activation of the cluster for wider or narrower set of input values. Thus criterion of resonance for each of the clusters can be different. This allows selective activation of clusters, yielding a map similar to Self Organizing Maps. Details of various implementations of this type of network are presented separately.

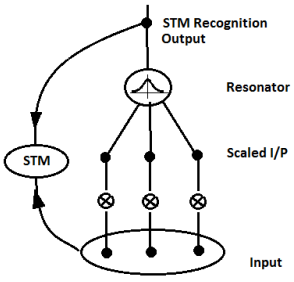

Figure 5: New ART2 structure

One of the basic equations used to implement a resonance condition is d(1-d), where 0<d<1. The resonance curve has a peak at d = 0.5 with peak value of 0.25. Therefore, the resonator uses the equation $\Sigma (d_i*(1-d_i))/n*4$, i = 1..n, where n is the number of input nodes and d_i indicates i^{th} input. Maximum value of this equation is 1. So, when a node is added, the inputs are scaled to generate d = 0.5, the node is in resonance condition. Figure 6 shows a resonance curve using this equation. Several other similar equations can also be used. Non-linear scaling can be used to sharpen or diffuse the resonance condition.

International Journal of Technology and Science, ISSN (Online) 2350-1111, (Print) 2350-1103 Volume 3, Issue 3, 2016, pp. 9-12

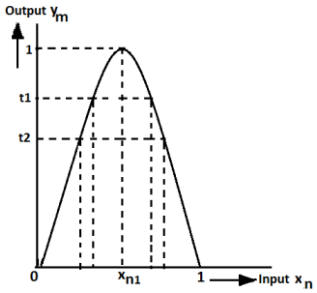

Figure. 6. Resonance in New ART2 model

This model is implemented for i number of inputs and j number of output clusters. User needs to specify i, n and t, where,

n is the number of iterations.

t is threshold.

Figure 7 shows the output of new ART2 network with two different thresholds. Following parameters are considered as an illustration,

$i = 1, j = 1, n = 50$

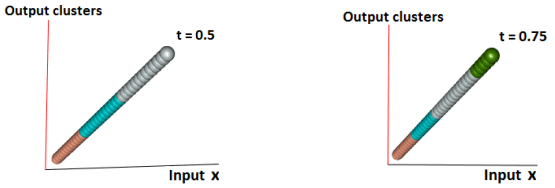

Figure 7: Output of new ART2, (a) t = 0.5 (b) t= 0.75

Figure 7(a) shows the result for a threshold of 0.5 in which number of clusters created at the output is 3. Whereas, Figure 7(b) shows the result for a threshold of 0.75 in which number of clusters created at the output is 4 for the same data set. It is observed that as the threshold increases the number of clusters created at the output also increases. In other words, increasing threshold results in narrowing of the input range. Here also group of circles of same color represents a single output cluster.

Figure 8 shows the output of new ART2 networt for an input of length 3. Following parameters are considered as an illustration,

$i = 3, j = 1, t = 0.7, n = 50$

Inputs are taken from 0:1. Inputs are classified into six clusters. Here also group of circles of same color represents a single output cluster. A new node is added when maximum output value of all nodes in the recognition layer is below a threshold 0.7.

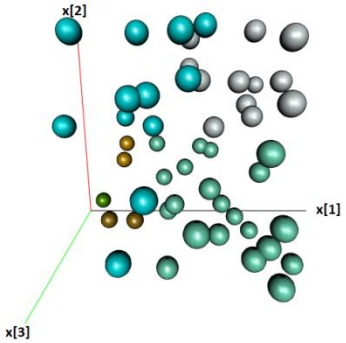

Figure 8: Output of new ART2

CONCLUSIONS

We have implemented a small library of functions in R, which can be used to simulate some of the commonly used ANNs. The library also includes some new ANN configurations defined by us, not available elsewhere. Building our own library allows comparative study of algorithms for implementing ANNs. We have presented some illustrative examples of training some of these ANNs in the paper. We intend to publish the code under GPL for the benefit of R users.

REFERENCES

[1] An Introduction to R, Notes on R: A programming Environment for Data Analysis and Graphics, Version 3.2.1 (2015-06-18)

[2] https://en.wikipedia.org/wiki/MATLAB

[3] Tom Mitchell, "Artificial Neural Networks", "Machine Learning", McGrawHill, pp. 81-127.

[4] A. Von Lehmen, E. G. Paek, P. F. Liao, A. Marrakchi and J. S. Patel, "Factors influencing learning by backpropagation," Proceedings of the IEEE International Conference on Neural Networks, Vol. I, pp. 335-341, 1988.

[5] Laurene Fausett, "Adaptive Resonance Theory", "Fundamentals Of Neural Networks", Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1994, pp. 218-288.

[6] Carpenter G. A., and Grossberg S, "ART2:Self organization of stable category recognition codes for analog input patterns", Applied Optics, vol. 26, pp. 4919-4930, 1987.

[7] Carpenter G. A., and Grossberg S., "ART2-A: An adaptive resonance algorithm for rapid category learning and recognition," Neural Network, vol. 4, pp. 493-504, Apr. 1991.

[8] C. Wang and J.C. Principe, "Training neural networks with additive noise in the desired signal," IEEE Transactions on Neural Networks, Vol. 10, No. 6, pp. 1511-1517,1999.

[9] G. Wang. Z. Tang, "A Modified Error Function for the Backpropagation Algorithm." Neurocomputing, Vol. 57, No. 3, pp. 477-484, 2004.

[10] Jiaoyan, Brian Funt and Lilong, " A New Type of ART2 Architecture and Application to Color Image Segmentation", Springer, ICANN 2008, Part I, LNCS 5163, pp. 89–98, 2008.

[11] Jian Fan, YangSong, MinRuiFei , "ART2 neural network interacting with environment", Elsevier, Neurocomputing , 72, pp 170–176, 2008.

[12] Bekir Karlik and A. Vehbi Olgac, "Performance Analysis of Various Activation Functions in Generalized MLP Architectures of Neural Networks", International Journal of Artificial Intelligence And Expert Systems (IJAE), Volume (1), Issue (4), pp 111-122, 2010.