Implementation of RSA with Feed-forward Neural Network using MATLAB

Somesh Kumar
Noida Institute of Engineering & Technology, Greater Noida

Rajkumar Goel
Noida Institute of Engineering & Technology, Greater Noida

ABSTRACT
In this paper the RSA algorithm has been implemented with feed forward artificial neural network using MATLAB. This implementation is focused on the network parameters like topology, training algoritahm, no. of hidden layers, no. of neurons in each layer and learning rate in order to get the more efficient results. Many examples are tested and it is obtained that two hidden layers feed forward neural network architectures will lead to optimal solution. Our goal in this paper is to obtain the minimum training time and minimum number of training iterations using the proposed optimal solution.

Keywords
RSA, Neural Network

1. INTRODUCTION
Artificial Neural Network (ANN) is a mathematical model designed to train, visualize, and validate neural network models[1].We can define the neural network model as a data structure that can be adjusted to produce a mapping from a given set of input and output data or relationships among the data. More specifically, the Neural Networks model uses numerical data to specify and evaluate artificial neural network models. This involves three basic steps. First, a neural network structure is chosen that is considered suitable for the type of data and underlying process to be modeled. Second, the neural network is trained by using a sufficiently representative set of data. Third, the trained network is tested with different data, from the same or related sources, to validate that the mapping is of acceptable quality.

There are many different types of ANN and techniques for training them but we are just going to focus on the three training algorithms: gradient descent with momentum and adaptive learning rate back propagation (traingdx), resilient back propagation (trainrp) and levenberg –marquardt backpropagation (trainlm), which are different variations of standard back propagation algorithm. Back propagation is by far the most widely used and understood neural network paradigm [2-6]. Its popularity arises from its simple architecture and easy to understand learning process, the back propagation scheme consists of two major steps. These are the forward activation and the backward error flows.

The training process begins with the assignment of random weights to the connections between the nodes of the various layers. The various input patterns are then presented to the network, and the forward activation flow produces the output patterns. These output patterns will not be the same as the desired output patterns. The errors in the outputs are calculated for the output layer nodes as the difference between the desired and actual outputs. For the hidden layers, the errors are calculated by back propagating the errors in the output layer to the hidden layers. The errors of each of the nodes are summed over the whole set of training patterns. These errors are used to change the weights in the interconnections between the layers. The weights connecting to the output layer are changed according to the delta rule, whereas for the weights in the hidden layers the generalized delta rule is used. There are many good references which describe the mathematics of the back propagation approach in detail including. The rest of the paper is organized as follows. In section-2 we describe the experimental setup for modeling the problem and training the neural network. Section 3 analyses the results taken from different experiments and at the last the observations and future works explained.

2. EXPERIMENTAL SETUP FOR TRAINING AND TESTING THE NEURAL NETWORK
Work has been started by finding the different values of encrypted strings corresponding to different values of plain text for different values of N. For this a no. of samples has been used. In the first experiment a string with 4 different symbols (@, A, B & C) has been used. Then the complete string is divided in the block size of 3 characters each and then applied the concept of number system to convert the incoming string into numeric values, with the concept:

@-0 A-1 B-2 C-3and Base of this system is 4.

eg if input string is C@ABC@ACCBCB

C@A	BC@
$3*4^2 + 0*4^1 + 1* 4^0$	$2*4^2 + 3*4^1 + 0* 4^0$
49	44
ACC	BCB
$1*4^2 + 3*4^1 + 3* 4^0$	$2*4^2 + 3*4^1 + 2* 4^0$
31	46

So the string will become

49 44 31 46

Then the above numeric string has been converted into the encrypted string using standard RSA algorithm.

All the above steps are implemented through a JAVA program which uses different user defined functions-

1. **static void inputp() throws IOException** To take input in string form and to break that strig in blocks of 3 character each and calculate equivalent numeric value.
2. **static void encrypt ()** To Encrypt the input string(in numeric form).

3. **static void converte() throws IOException** To convert the encrypted numeric values in character form.
4. **static void decrypt() throws IOException** To decrypt the encrypted numeric values.
5. **static void convertd() throws IOException** To convert the decrypted numeric values in string form.
6. **static long mod(long z,int ed)** To calculate (z power ed) mod n

In this way different sample values for different number of symbols having a maximum value = 63 i.e n=64 have been collected.

These values are the inputs and corresponding encrypted string (In numeric form) are the target outputs of the ANN for performance evaluation on the basis of different factors.

This experiment is done for different values of N (e.g n,=64 and n=133) and different number of symbols in the plain text [7-8].

Here the network is having a single neuron in input layer, a hidden layer with 9 neurons and one neuron in the output layer. Like this, we have created the network with different topologies with different number of neurons in hidden layers having different training methods and different learning rates.

***Network architecture*:** The definition of "optimal" network architecture for any particular problem is quite difficult and remains an open problem. To this end we tested a variety of topologies with different number of hidden layers and with various numbers of neurons at each layer. The results reported are the best results obtained for each problem.

***Normalization*:** To make the adaptation of the network easier, the data are transformed through the normalization procedure that takes place right before training. Assuming that the data presented to the network are in Z*p*, where *p* is prime, the space $S = [-1, 1]$, is split in *p* subspaces. Thus, numbers in the data are transformed to analogous ones in the space *S*. At the same time, the network output is transformed to a number within Z*p* using the inverse operation.

3. RESULTS ANALYSIS

Analysis is done by varying the number of hidden layers, number of neurons in hidden layers, learning rate and by using different training methods. To test the network performance two different measures are considered. The first measure is called as *complete measure* and denoted by μ_0, indicating the percentage of training data, for which the network is able to compute the exact target value. This is not sufficient enough as the network performance indicator. The fact that the network output is restricted within the range (-1, 1) plays a significant role. Very small differences in output render the network unable to find the exact target but to be very close to it. So, using as a second measure, called as *near measure* and denoted by μ_v the percentage of the data for which the difference between desired and actual output does not exceed ±v of the real target, gives a better understanding of the network performance. It is also made clear that the second measure shows the real capability of the network. Since when the training procedure is continued long enough the first measure kept rising to reach the second one. It is important to be mentioned that the *near measure* μ_v plays a significant role.

The efficiency of neural network with 1 hidden layer and 2 hidden layers having different no. of neurons is represented in Tables from 1 to 8. Three different variations of back propagation algorithm used to train the neural network are gradient descent with momentum and adaptive learning rate back propagation (traingdx), resilient back propagation (trainrp) and levenberg-marquardt back propagation (trainlm). All these results are obtained by taking 64 and 133 samples.

Table 1: Efficiency of neural network with 1 hidden layer having different number of neurons, on the basis of different training methods (For number of samples =64).

Training Method → / Topology ↓	Traingdx				Trainrp				Trainlm			
	μ_0	μ_1	μ_2	μ_3	μ_0	μ_1	μ_2	μ_3	μ_0	μ_1	μ_2	μ_3
1-13-1	3	13	21	48	22	37	51	72	13	38	46	68
1-14-1	3	13	23	50	23	37	52	72	19	44	51	68
1-15-1	2	13	23	54	23	38	50	74	23	45	53	68
1-16-1	2	14	27	58	22	38	51	73	27	47	57	68
1-17-1	2	14	35	64	26	40	53	76	32	51	60	70
1-18-1	2	14	34	66	33	46	57	78	37	57	66	74
1-19-1	2	13	32	66	37	51	63	79	41	61	73	77
1-20-1	2	13	32	70	46	60	68	81	49	65	78	81

Table 2: Efficiency of neural network with 2 hidden layer having different number of neurons, on the basis of different training methods (for number of samples =64).

Training Method → / Topology ↓	Traingdx				Trainrp				Trainlm			
	μ_0	μ_1	μ_2	μ_3	μ_0	μ_1	μ_2	μ_3	μ_0	μ_1	μ_2	μ_3
1-5-7-1	2	22	25	52	11	41	43	48	64	71	79	81
1-6-8-1	2	22	28	53	13	41	43	53	70	82	82	84
1-7-9-1	3	24	30	56	13	43	48	59	70	86	91	92
1-8-10-1	3	26	34	58	17	47	54	70	91	98	100	100
1-9-11-1	2	27	40	64	21	56	68	81	98	100	100	100
1-10-12-1	2	33	48	68	21	55	64	81	98	100	100	100
1-11-13-1	3	43	54	78	22	53	59	81	100	100	100	100
1-5-7-1	2	22	25	52	11	41	43	48	64	71	79	81

Table 3: Efficiency of neural network with 1 hidden layer having different number of neurons, on the basis of different training methods (For number of samples =133).

Training Method → / Topology ↓	Traingdx				Trainrp				Trainlm			
	μ_0	μ_1	μ_2	μ_3	μ_0	μ_1	μ_2	μ_3	μ_0	μ_1	μ_2	μ_3
1-20-1	1	8	15	23	3	15	23	32	11	28	32	43
1-21-1	1	10	15	23	3	15	23	34	11	28	32	48
1-22-1	1	10	15	23	3	15	25	36	13	31	34	49
1-23-1	1	11	15	25	4	15	27	36	15	36	45	53
1-24-1	2	11	15	25	4	16	27	38	15	38	45	55
1-25-1	2	11	17	27	5	16	28	38	17	38	48	55
1-26-1	2	11	17	27	7	19	31	39	23	42	49	59
1-20-1	1	8	15	23	3	15	23	32	11	28	32	43

Table 4: Efficiency of neural network with 2 hidden layers having different number of neurons, on the basis of different training methods (For number of samples =133).

Training Method → / Topology ↓	Traingdx				Trainrp				Trainlm			
	μ_0	μ_1	μ_2	μ_3	μ_0	μ_1	μ_2	μ_3	μ_0	μ_1	μ_2	μ_3
1-11-13-1	2	28	35	60	19	39	41	59	69	87	90	94
1-12-14-1	3	28	37	63	19	47	49	61	71	87	92	94
1-13-15-1	5	28	43	63	19	49	56	66	81	92	92	98
1-14-16-1	7	29	45	64	19	59	69	77	98	99	100	100
1-15-17-1	7	35	46	78	21	59	69	77	98	99	100	100
1-16-18-1	8	37	49	80	23	62	71	77	98	100	100	100
1-17-19-1	8	45	56	80	26	62	71	78	98	100	100	100
1-11-13-1	2	28	35	60	19	39	41	59	69	87	90	94

Table 5: Efficiency of neural network with 1hidden layer having different number of neurons, on the basis of different Learning rate (For number of samples =64).

Learning Rate → / Topology ↓	η_1(0.05)				η_2(0.005)				η_3(0.0005)			
	μ_0	μ_1	μ_2	μ_3	μ_0	μ_1	μ_2	μ_3	μ_0	μ_1	μ_2	μ_3
1-13-1	21	38	49	62	13	38	46	68	11	40	51	65
1-14-1	27	44	51	64	19	44	51	68	16	47	57	71
1-15-1	33	47	51	64	23	45	53	68	23	54	63	73
1-16-1	41	55	55	65	27	47	57	68	31	59	69	78
1-17-1	48	59	60	67	32	51	60	70	40	65	75	84
1-18-1	49	63	63	71	37	57	66	74	47	64	75	83
1-19-1	51	67	66	77	41	61	73	77	52	64	74	84
1-20-1	51	68	75	86	49	65	78	81	56	67	72	84

Table 6: Efficiency of neural network with 2 hidden layers having different number of neurons, on the basis of different Learning rate (For number of samples =64).

Learning Rate → / Topology ↓	η_1(0.05)				η_2(0.005)				η_3(0.0005)			
	μ_0	μ_1	μ_2	μ_3	μ_0	μ_1	μ_2	μ_3	μ_0	μ_1	μ_2	μ_3
1-5-7-1	64	78	78	86	64	71	79	81	41	61	81	81
1-6-8-1	71	84	84	86	70	82	82	84	47	61	81	85
1-7-9-1	71	84	86	91	70	86	91	92	56	78	86	95
1-8-10-1	82	90	92	98	91	98	100	100	98	98	98	100
1-9-11-1	98	98	98	100	98	100	100	100	100	100	100	100
1-10-12-1	98	98	100	100	98	100	100	100	100	100	100	100
1-11-13-1	100	100	100	100	100	100	100	100	100	100	100	100
1-5-7-1	64	78	78	86	64	71	79	81	41	61	81	81

Table 7: Efficiency of neural network with 1hidden layer having different number of neurons, on the basis of different Learning rate (For number of samples =133).

Learning Rate → / Topology ↓	η_1(0.05)				η_2(0.005)				η_3(0.0005)			
	μ_0	μ_1	μ_2	μ_3	μ_0	μ_1	μ_2	μ_3	μ_0	μ_1	μ_2	μ_3
1-20-1	11	32	36	43	11	28	32	43	12	38	40	48
1-21-1	11	32	36	44	11	28	32	48	12	38	40	49
1-22-1	11	32	38	44	13	31	34	49	13	38	41	53
1-23-1	13	32	38	50	15	36	45	53	13	39	43	54
1-24-1	13	32	38	53	15	38	45	55	17	39	43	54
1-25-1	14	34	39	54	17	38	48	55	19	41	47	55
1-26-1	14	34	39	57	23	42	49	59	21	41	48	55
1-20-1	11	32	36	43	11	28	32	43	12	38	40	48

Table 8: Efficiency of neural network with 2 hidden layers having different number of neurons, on the basis of different Learning rate (for number of samples =133).

Learning Rate → / Topology ↓	η_1(0.05)				η_2(0.005)				η_3(0.0005)			
	μ_0	μ_1	μ_2	μ_3	μ_0	μ_1	μ_2	μ_3	μ_0	μ_1	μ_2	μ_3
1-11-13-1	63	78	81	83	69	87	90	94	62	69	69	74
1-12-14-1	69	78	83	87	71	87	92	94	78	81	87	92
1-13-15-1	78	81	87	92	81	92	92	98	87	92	98	98
1-14-16-1	92	98	98	100	98	99	100	100	98	100	100	100
1-15-17-1	92	98	98	100	98	99	100	100	98	100	100	100
1-16-18-1	98	98	99	100	98	100	100	100	99	100	100	100
1-17-19-1	98	100	100	100	98	100	100	100	100	100	100	100
1-11-13-1	63	78	81	83	69	87	90	94	62	69	69	74

4. CONCLUSION AND FUTURE SCOPE

In this work neural network approach has been used to encounter the problem in RSA algorithm. Our observation with this attempt is that it is possible to train feed forward neural networks to tackle this task with the help of feed forward neural networks. In general, the problem can be solved if the architectural topology is small enough. Here, with a predetermined number of trial and error procedures the problem can be resolved. Very small prime numbers are selected in order to have an extensive study related to network's architecture and their ability for various efficient training methods. All three methods have been extensively tested with a wide range of parameters. From the experimental results shown in the Tables 1-8, it has been observed that the training method and learning rate does not play a significant role in tackling the particular problem. On the other hand a crucial role is being played by the network architecture and the normalization portion of the training algorithm used. An efficient system can be achieved by focusing on the better architecture. Focusing only on number of neurons will increases the complexity and by increasing the number of layers unnecessarily will increases the processing time of a network.

Here we have considered only feed-forward artificial neural networks, in the future we intend to apply various other neural networks and learning techniques such as bidirectional neural networks with different learning methods, and radial basis function networks. We also intend to apply samples greater than 1000 and more training algorithms.

5. REFERENCES

[1] Ibrahim Subariah and Maarof Mohd Aizaini, "*A Review on Biological Inspired Computation in Cryptology, Jurnal Teknologi Maklumat", Journal of Information Technology*, Vol. 17, no. 1, pp 90-98, (2007).

[2] Ciampi Antonio and Zhang Fulin, "A new approach to training back-propagation artificial neural networks: empirical evaluation on ten data sets from clinical studies", *Statistics in Medicine*, Vol.21, Issue-9, pp 1309-1330, (2002).

[3] Hagan Martin T. and Menhaj Mohammad B., "Training feed forward networks with the Marquardt Algorithm", IEEE Transactions on Neural Networks, Vol. 5, no. 6, pp 989-993, (1994).

[4] Bhavsar Hetal and Ganatra Amit, "A comparative study of training algorithms for supervised machine learning", *International Journal of Soft Computing and Engineering*, Vol. 2, Issue-4, pp 74-81, (2012)

[5] Istook Ernest and Martinez Tony, "Improved backpropagation learning in neural networks with windowed momentum", *International Journal of Neural Systems*, Vol. 12, Issue 3&4, pp 303-318, (2002)

[6] Laskari, E.C., Meletiou, G.C., Tasoulis, D.K. and Vrahat, M.N., "Studying the performance of artificial neural networks on problems related to cryptography", *Nonlinear Analysis: Real World Applications*, Vol.7, Issue 5, pp 937-942, (2009).

[7] RSA Laboratories, Why RSA? Available at: http://www.rsa.com/ rsalabs/node.asp?id=2222 and http://www.rsa.com/rsalabs/ node.asp?id=2223.

[8] Vishwakarma Virendra P. and Gupta M. N., "A New Learning Algorithm for Single Hidden Layer Feed forward Neural Networks", *International Journal of Computer Applications*,Vol. 28, no.6, pp 26-33, (2011)